


45

Le journal qui se bat pour ses lecteurs

MAI 1988

# DBCOOK: TURBO-Forth marche sur les pieds d'ASHTON-TATE.

Va-t-il se laisser faire ?

## EDITORIAL

Serions-nous en train de marcher sur les pieds d'ASHTON-TATE? Aujourd'hui une extension permettant de gerer les fichiers dBASE III/III+ depuis TURBO-Forth, demain un micro-processeur spécialisé en SGBD, programmable en Forth et de fabrication française! C'est en tout cas ce qui est proposé dans ce numero de JEDI. Et tant qu'à marcher sur les pieds des autres, j'apprends que dans FORTH DIMENSION, ils ont pondu un programme accédant aux fichiers dBASE, mais bien moins évolué que le nôtre. Arghh!! FORTH DIMENSION baisserait-il sa garde? L'élève dépasserai-t-il le maitre? Sommes-nous surdoués? Avons-nous mangé du TOPSET? Sue Ellen se réconciliera-t-elle une fois de plus avec JR?

Et même que nous avons ouvert notre FORUM sur 3615 SAM*JEDI. Et il y en a qui ne l'ont pas encore essayé. Mais à quoi ça sert que nous nous décarcassions chez JEDI? Pimentez vos relations avec un zeste de TELETEL, c'est ça une association dynamique et évoluée. Des questions? Des réponses? Des problèmes avec FORTH, TURBO-Forth, Volks-FORTH, autre-chose-Forth? Un seul réflexe, 3615 SAM*JEDI: jamais encombré, disponible 24h/24, 365.25 jours/ans, 32 voies d'accès simultanées... Même les américains restent muets.

## SOMMAIRE






Nos coordonnées: ASSOCIATION JEDI, 17, rue de la Lancette 75012 PARIS
tel président: (1) 43.40.96.53
tel secrétaire: (1) 49.85.63.67 (nouveau n° de tel)
BAL secrétaire: 3615 SAM*JEDI SECRETAIRE


Le serveur SAM est la propriété exclusive de la Sté VICTEL, 14 bd Montmartre, 75009 PARIS, tel: 45.23.02.79.

## TRITURATIONS BINAIRES

par Marc PETREMANN

FORTH 83-Standard: pour tous systèmes CP/M, MSDOS, VolksFORTH ATARI, etc...

Les nombres ont parfois de bien curieuses propriétés quand on les réarrange selon des dispositions particulières. Dans un précédent article au sujet de la logique binaire, j'avais mentionné le codage binaire classique et celui dénommé code binaire corrigé, appelé aussi code AIKEN:

| codage binaire ordinaire | AIKEN |
|---|---|
| 000 0 | 000 0 |
| 001 1 | 001 1 |
| 010 2 | 011 3 |
| 011 3 | 010 2 |
| 100 4 | 110 6 |
| 101 5 | 111 7 |
| 110 6 | 101 5 |
| 111 7 | 100 4 |

Le code binaire dont la disposition a été réarrangé dans la colonne AIKEN possède une première propriété:" les nombres consécutifs n'ont qu'un seul digit variant avec le précédent ou le suivant". Mieux, les extrémités de cette série peuvent être reliées sans que le précédent énoncé ne soit mis en défaut.

Dans notre exemple, la série a été développée à partir de 2E3 éléments, mais elle peut l'être pour un nombre quelconque d'éléments, ledit nombre étant toujours une puissance de deux. Les éléments contenus dans la série binaire AIKEN sont situés entre 0 et 2EN-1 pour 2EN éléments: le nombre de digits pris en compte est égal à N.

A priori, le réarrangement dans l'ordre binaire AIKEN n'a aucune particularité en décimal. Pourtant, en y regardant de plus près, il ressort de la disposition suivante:

0 1 3 2 6 7 5 4

qu'en effectuant les différences entre les éléments indices:

n(2EN-i)-n(i)

pour i compris entre 1 et 2EN/2, on obtient toujours une valeur constante:

```
pour i=1  n(2EN-1)=4  n(1)=0  4-0=4
pour i=2  n(2EN-2)=5  n(2)=1  5-1=4
pour i=3  n(2EN-3)=7  n(3)=3  7-3=4
pour i=4  n(2EN-4)=6  n(4)=2  6-2=4
```

Cette règle s'applique également à une série plus grande.

Certes, en grattant un peu, pourra-t-on dégager une règle mathématique permettant de retrouver la valeur de l'élément d'indice i d'une suite binaire AIKEN d'ordre 2EN pour N étant le nombre de digits binaires pris en comptes. Exemple, quelle est la valeur de l'élément d'indice 23 d'une série d'ordre 2E8, c'est à dire de 256 éléments codés sur huit bits? A ce jour, je n'ai pas trouvé de méthode de calcul, mais seulement une méthode de découpage géométrique de la série d'ordre 2EN. Je propose aux matheux de résoudre ce petit problème.

Maintenant, repartons de notre série binaire classique d'ordre 2E3 ayant servi à l'illustration du précédent propos:

| série binaire classique | réfléchie |
|---|---|
| 000 0 | 000 0 |
| 001 1 | 100 4 |
| 010 2 | 010 2 |
| 011 3 | 110 6 |
| 100 4 | 001 1 |
| 101 5 | 101 5 |
| 110 6 | 011 3 |
| 111 7 | 111 7 |

La série binaire nommée "réfléchie" reprend "en miroir" les digits binaires de la série classique: on a "retourné" comme une chaussette les N premiers bits de la série d'ordre 2EN. Ici, l'algorithme a pu être déterminé et réalisé en FORTH:

```
: 2EN  ( n --- 2en)
 DUP IF 1 SWAP 0
       DO  2 * LOOP
    ELSE DROP 1
    THEN ;
: REFL  ( n nbits --- réfléchi de n sur n bits)
 DUP >R 1- 0 DO 2 /MOD LOOP
 0 0 R> SWAP
 DO  SWAP
     IF   I 2PUISS +
     THEN
 LOOP ;
```

Exemple:

```
 0 3 REFL .    affiche 0
 1 3 REFL .    affiche 4
 2 3 REFL .    affiche 2
etc...
 7 3 REFL .    affiche 7
```

Ici, nouvelle surprise, la série développée par cette méthode a aussi ses propriétés particulières:

- de part et d'autre de la ligne de symétrie située entre les éléments d'indice 2EN/2 et 2EN/2+1, les nombres sont d'abord tous pairs, puis ensuite tous impairs. En fait, quand on regarde les équivalents binaires de la série d'origine, la cause de cette propriété est évidente: elle dépend du bit de poids fort qui devient le bit de poids faible en binaire réfléchi.

- les sommes deux à deux des éléments d'indice 2EN-i et i donnent toujours une constante égale à 2EN-1, pour i compris entre 1 et 2EN/2:

0 4 2 6 1 5 3 7

```
pour i=1  n(2EN-1)=7  n(1)=0  7+0=7
pour i=2  n(2EN-2)=3  n(2)=4  3+4=7
pour i=3  n(2EN-3)=5  n(3)=2  5+2=7
pour i=4  n(2EN-4)=1  n(4)=6  1+6=7
```

- les éléments d'indice 2EN/2+1 à 2EN correspondent à la somme des éléments respectifs d'indice 1 à 2EN/2 augmentés de 1.

Voilà que le code binaire réfléchi donne à réfléchir. Je vous laisse le soin de développer et formuler une règle s'appliquant à tous les cas. ■

FORTH

## Import-export dBASE Turbo-FORTH-83
V 1.0 01/06/88

M. Zupan & Ass. JEDI

Système: TURBO-Forth 83-Standard MSDOS
Diffusion: fichier DBCOOK.FTH sur 3615 SAM*JEDI en téléchargement; prochainement dans le module M4 de TURBO-Forth.
Adaptabilité: très difficile à d'autres systèmes

LISTING: DBCOOK.FTH

```
only definitions forth vocabulary DATABASE
also database definitions decimal
```

```
\ DESCRIPTION D'UN FICHIER DE DONNEES dBASE
\ -----------------------------------------
-1024 LBUF FILES 6 + * -
constant  LIMIT
  \ limite haute des tampons du Forth
variable  DBFILE
  \ handle du fichier .dbf courant
64 String DBFILE$
  \ chemin du fichier .dbf courant
Create    DBTOP 18 allot
  \ table principale du fichier courant
: CAPACITY   ( -- u )   dbtop  4 + @ ;
  \ nombre d'enregistrements
: B/HEADER   ( -- n )   dbtop  8 + @ ;
  \ longueur de l'en-tete fichier
: B/RECORD   ( -- n )   dbtop 10 + @ ;
  \ longueur d'un enregistrement
: #FIELDS    ( -- n )   dbtop 12 + @ ;
  \ nombre de champs par enregistrement
: FLD-BUFFER ( -- adr ) dbtop 14 + @ ;
  \ tampon des descripteurs de champs
: REC-BUFFER ( -- adr ) dbtop 16 + @ ;
  \ tampon d'enregistrement
variable  REC
  \ enregistrement courant
 : RECORD ( -- n ) rec @ ;
variable  FLD
  \ champ courant
 : FLD!   ( n -- ) fld ! ;

\ DESCRIPTION DES CHAMPS
\ ----------------------
: (FLD)    ( -- adr )
  \ descripteur du champ courant
  fld @ 1- 20 * fld-buffer + ;
: FLDNAME ( -- adr len )
  \ nom du champ
  (fld) dup 11 0
  do count 0= ?Leave loop
  over - 1- ;
: FLDTYPE ( -- c )  (fld) 11 + c@ ;
  \ type du champ (C,N,L,M,D)
: FLDPOS  ( -- adr )
  \ position du champ dans le record
  (fld) 12 + @ fld-buffer
  12 + @ - 1+ rec-buffer + ;
: FLDLEN  ( -- n )  (fld) 16 + c@ ;
  \ longueur du champ
: FLDDEC  ( -- n )  (fld) 17 + c@ ;
  \ nombre de décimales
: FLDFLAG ( -- fl ) (fld) 18 + @  ;
  \ flag de sélection d'un champ

\ GESTION DE LA DATE COURANTE
\ ---------------------------
code (DATE)  ( -- aa mj )
  \ récupère la date MS-DOS
42 # ah mov 33 int cx push dx push next end-code
: DATE!       ( -- )
  \ met date à jour dans header
  (date) flip swap 1900 - dbtop 1+ tuck c! 1+ ! ;

\ LECTURES/ECRITURES ACCES DIRECT DU FICHIER
\ ------------------------------------------
Variable  UPDATE
  \ drapeau enregistrement modifié

: READ-TOP     ( -- )
  \ charge l'en-tête
  0 0 dbfile @ 0 (seek) ?Dos-err 2drop
  dsegment dbtop 12 dbfile @ (get) ?Dos-err drop
  b/header 32 / 1-        * dbtop 12 + !
  ( nombre de champs )
  limit #fields 20 * -      dbtop 14 + !
  ( tampon descripteur champs )
  fld-buffer b/record 1+ - dbtop 16 + !
  ( tampon d'enregistrement )
  26 fld-buffer 1- c! ;
  ( ctrl-Z fin de fichier )

: WRITE-TOP    ( -- )
  \ re-écrit l'en-tête
  date! 0 0 dbfile @ 0 (seek) ?Dos-err 2drop
  dsegment dbtop 12 dbfile @ (put) ?Dos-err drop ;

: READ-FIELDS  ( -- )
  \ lit les descripteurs de champs
  #fields 0 do
  32 I 1+ * s>d dbfile @
  0 (seek) ?Dos-err 2drop
  dsegment fld-buffer I 20 * + 18
  dbfile @ (get) ?Dos-err drop
  loop ;

: SEEK-RECORD  ( n -- )
  \ pointe l'enregistrement n
  dup 0= over capacity u> or
  abort" enregistrement hors limites"
  1- b/record um* b/header s>d d+
  dbfile @ 0 (seek) ?Dos-err 2drop ;

: READ-RECORD  ( -- )
  \ lit l'enregistrement courant
  record seek-record
  dsegment rec-buffer
  b/record dbfile @ (get) ?Dos-err drop ;

: WRITE-RECORD ( -- )
  \ sauve l'enregistrement courant
  update @
  if record seek-record
     dsegment rec-buffer
     b/record dbfile @ (put) ?Dos-err drop
     update off write-top
  then ;

: APPEND-BLANK ( -- )
  \ ajoute un enregistrement vide
  write-record
  rec-buffer b/record blank
  dbtop 4 + 1+!
  capacity seek-record
  dsegment rec-buffer
  b/record 1+ dbfile @ (put) ?Dos-err
  b/record <= abort" disque plein"
  write-top capacity rec ! ;
  ( = enregistrement courant )

\ CREATION DES MOTS-CHAMPS DANS LE DICTIONNAIRE FORTH
\ --------------------------------------------------
: ()CREATE        ( -- )
  \ crée le mot-champ courant
  fldname here place
  here count upper
  here "create fld @ ,
  does> @ fld! ;
  \ à l'exécution: champ courant

: CREATE-FIELDS ( -- )
  \ multi-création des mots-champs
  #fields 1+ 1
  do I fld! ()create loop ;

\ SELECTION DES CHAMPS ACTIFS
\ ---------------------------
: SET          ( -- )
  \ sélectionne le champ courant
  (fld) 18 +  on ;
: EXCEPT     ( -- )
  \ dé-sélectionne le champ courant
  (fld) 18 + off ;
: SELECTION ( -- )
  \ prépare une sélection de champs
  #fields 1+ 1 do I fld! except loop ;
: FIELDS     ( -- )
  \ sélectionne tous les champs
  #fields 1+ 1 do I fld! set     loop ;

\ OPERATEURS CHAINES SUR LE CHAMP COURANT
\ ---------------------------------------
: ()          ( -- adr len )
  \ champ courant
  fldpos fldlen ;

: -TRAIL-  ( adr len -- adr' len' )
  \ suppression espaces début et fin
```

```
  -trailing bl skip ;

: ()DVAL    ( -- d )
  \ conversion champ -> nbre double
  base @ >r decimal
  () -trail- pad place
  bl pad count + c!
  pad number?
  ( position du point décimal récupérable dans DPL )
  0= if 2drop 0 0 then r> base ! ;

: COMPARE$ ( ad1 ln1 ad2 ln2 -- fl )
  \ Compare 2 chaines d'inégale longueur
  -trail- 2swap -trail-
  rot 2swap 2over min compare ?dup
  if -rot 2drop
  else 2dup <> -rot > ?negate then ;   \ fl = 0 -1 ou 1
  \ COMPARE$ est plus pratique et plus général  que COMPARE
 (ad1 ad2 l --fl)
  \ et comme pour lui, les minuscules sont
  \ converties selon CAPS

: ()=       ( adr len -- fl )
  \ test chaine=champ courant
  () compare$ 0= ;
: ()>=      ( adr len -- fl )
  \ test chaine>=champ
  () compare$ 0>= ;
: ()<=      ( adr len -- fl )
  \ test chaine<=champ
  () compare$ 0<= ;
: ()$       ( adr len -- fl )
  \ test présence de sous-chaine
  () search nip ;

' u<= ' search 12 + !
( cette ligne corrige le SEARCH)
( des premières versions Turbo-Forth )
( elle peut être supprimée dans les versions déboguées )

\ DEPLACEMENTS DANS LE FICHIER
\ ---------------------------
: GO           ( n -- )
  \ positionne record courant
  write-record dup record <>
  if rec ! read-record else drop then ;

: COPY         ( n -- )
  \ copie record courant dans record n
  rec ! update on write-record ;

: PERMUTE      ( n -- )
  \ permute record courant avec record n
  rec-buffer dup b/record tuck - swap cmove
  record over go copy
  rec-buffer dup b/record tuck - -rot cmove
  copy ;

\ PROCEDURES VECTORISEES CONDITIONNELLES
\ SUR L'ENSEMBLE DU FICHIER
\ ------------------------------------------
defer CONDITION ( -- flag )
  \ vecteur général de condition

: ALL            ( -- )
  \ supprime toute condition
  ['] true is condition ;
: (FOR)          ( -- )
  \ FOR en compilation
  r> dup 2+ >r @ is condition ;
: FOR            ( <mot_test> -- )
  \ active une condition
  state @ if compile (for)
        else ' is condition then ; immediate

variable CNT
  \ compteur
defer PROCESS   ( -- )
  \ Vecteur général d'action record

: DOPROCESS     ( -- )
  \ Applique PROCESS au fichier
  cnt off capacity 1+ 1
  do I go condition if process cnt 1+! then
     stop? ?Leave
  loop ;

: (PROCEED)     ( -- )
  \ Proceed en compilation
  r> dup 2+ >r @ is process doprocess ;

: PROCEED       ( <procedure> -- )
  \ Exécute une procédure fichier
  state @ if compile (proceed)
       else ' is process doprocess then ; immediate

: .COUNT         ( -- )
  \ affiche compteur de proceed
  cr cnt @ 6 u.R ."  enregistrement(s) " ;

\ AFFICHAGE DE LA STRUCTURE DU FICHIER COURANT
\ -----------------------------------------------
: .STRUCTURE    ( -- )
   cr ." Structure du fichier    : " dbfile$ type
   0 0 dbfile @ 2 (seek) ?Dos-err
   cr ." Taille en octets        : " 9 d.R
   cr ." Octets d'en-tête        : " b/header 9 .R
   cr ." Nombre d'enregistrements: " capacity 9 u.R
   cr ." Dernière mise à jour    :  "
   dbtop 1+ count 0 rot count swap c@ 100 * + 100 um* d+
   <# # # ascii / hold # # ascii / hold # # #> type
   cr ." Champ  nom champ   type        dim    déc"
   #fields 1+ 1
   do cr I 5 .R 2 spaces I fld!
        fldname type 19 ?Tab
        fldtype case ascii C of ." Caractère" endof
                     ascii N of ." Numérique" endof
                     ascii L of ." Logique"   endof
                     ascii D of ." Date"      endof
                     ascii M of ." Mémo"      endof
               endcase
        fldlen 32 ?Tab 3 .R 39 ?Tab
        flddec ?Dup if 3 .R then
     stop? ?Leave loop
  cr ." ** Total **" 30 ?Tab b/record 5 u.R cr ;

\ DEBUT ET FIN DE SESSION DE TRAVAIL SUR UN FICHIER dBASE
\ --------------------------------------------------------
: USE         ( <filename[.dbf]> -- )
  \ ouvre un fichier .DBF
  ?open  ext$ pad place " .DBF" ext$ $!
  \ ext. .DBF par défaut
  filename 2 (open)
  \ ouvre lecture/écriture
  here count dbfile$ $!
  \ conserve chemin\nom-fichier
  pad  count ext$    $!
  \ restitue extension usuelle
  ?Dos-err dbfile !
  \ handle du fichier .dbf
  read-top
  \ lit l'en-tête du fichier
  read-fields
  \ charge descripteurs de champs
  only forth also database definitions
  \ vocabulaire database de travail
  " mark SESSION" $execute
  \ début d'une session de travail
  create-fields
  \ crée les mots de champs
  rec off 1 fld! update off all fields
  \ initialise l'utilisation
 ;

: END       ( -- )
  \ clôture de session USE
  " forget SESSION" $execute cr
  \ oublie la session de travail
  write-record
  \ sauve dernier record si modifié
  dbfile @ (close)
  \ ferme fichier .dbf
  ." Clôture de " dbfile$ type cr ;
  \ confirme
```

```
\ DISPLAY FICHIER SELON CHAMPS SELECTIONNES ET CONDITION
\ --------------------------------------------------------
variable RECNOS recnos on
  \ flag display n° d'enregistrement
variable COLS
  \ nombre de colonnes en sortie
80 cols !
  \ pour l'ecran

: ?CRDISP   ( width -- )
  \ passage à la ligne dans display
  #out @ + cols @ 1- >=
  if cr recnos @
     if 7 else 2 then spaces
  then ;

: LDISP     ( adr len width -- )
  \ affichage formaté à gauche
  dup ?crdisp over - -rot type spaces ;

: RDISP     ( adr len width -- )
  \ affichage formaté à droite
  dup ?crdisp over - spaces type ;

: ()DISPLAY ( -- )
  \ Affiche champ courant
  () dup fldname nip max fldtype case
            ascii C of ldisp endof
            ascii N of rdisp endof
            ascii M of 4 max -rot 2drop
                       " mémo" rot ldisp endof
            ascii L of 3 max dup ?crdisp -rot
                       drop ascii . emit
                       c@ emit ascii . emit
                       3 - spaces endof
            ascii D of dup ?crdisp -rot drop
                       dup 6 + 2 type
                       ascii / emit dup 4 + 2 type
                       ascii / emit
                       2 + 2 type 8 - spaces endof drop
            endcase ;

: []DISPLAY ( -- )
  \ display record selon selection
  cr recnos @ if record 5 u.R then
  rec-buffer c@ emit space
  #fields 1+ 1 do I fld!
           fldflag if ()display space then loop ;

: DISPLAY   ( -- )
  \ display fichier selon condition
  cr bold recnos @ if ." Enr.N° " else 2 spaces then
  #fields 1+ 1 do I fld!
           fldflag if fldname fldlen over max
                      fldtype case
                      ascii C of ldisp endof
                      ascii N of rdisp endof
                      ascii L of 3 max ldisp endof
                      ascii M of 4 max ldisp endof
                      ascii D of ldisp endof endcase
                      space
                   then
           loop attoff
  proceed []display .count ;

\ SUPPRESSION D'ENREGISTREMENTS
\ -----------------------------
: DELETED   ( -- fl )
  \ record courant effacé ?
  rec-buffer c@ ascii * = ;
: UNDELETED ( -- fl )
  \ record courant valide ?
  deleted not ;

: []DELETE  ( -- )
  \ pré-efface record courant
  undeleted if
  ascii * rec-buffer c! update on then ;

: []RECALL  ( -- )
  \ restitue record courant
  deleted if
  bl rec-buffer c! update on then ;

: DELETE    ( -- )
  \ efface fichier selon condition
  proceed []delete .count ." effacé(s)" ;

: RECALL    ( -- )
  \ restitue fichier selon condition
  proceed []recall .count ." restitué(s)" ;

\ EDITIONS ET AJOUTS D'ENREGISTREMENTS
\ -----------------------------------
: ?NUM      ( adr len -- fl )
  \ test chaîne numérique
  base @ >r decimal pad place
  bl pad count + c!
  pad number? -rot 2drop
  if dpl @ dup
     if flddec ?dup
        if = else -1 = then
     then
  else false
  then r> base ! ;

: REPLACE   ( adr len -- )
  \ place une chaine dans le champ
  ?dup if
  () blank -trail-
  fldtype case
          ascii C of () rot min cmove endof
          ascii N of 2dup ?num
                     if () rot over min dup
                        >r - + r> cmove
                     else 2drop
                     then endof
          ascii L of drop c@ fldpos c! endof
          ascii D of 2dup ?num swap 8 = and
                     if () cmove
                     else drop
                     then endof
          ascii M of 2drop endof 2drop endcase
  update on
  else drop then ;

: ()EDIT    ( -- )
  \ édite champ courant
  cr fldname type 12 ?tab invers
  () type fldlen backspaces
  here fldlen expect
  span @ dup 1+ fldlen min backspaces
  here swap replace () type attoff ;

: []EDIT    ( -- )
  \ édite record selon sélection
  cr ." Enreg. n°" record u.
  #fields 1+ 1 do I fld!
          fldflag if ()edit then
        loop write-record ;

: RETRY?    ( -- fl1 fl2 )
  \ question fin d'édition record
  cr ." <R> reprend / <Q> quitte / continue " key upc
  case ascii R of false false endof
       ascii Q of true  true  endof
              false true rot endcase ;

: EDIT      ( -- )
  \ édite fichier selon condition
  capacity 1+ 1
  do I go false condition
     if begin drop []edit retry? until
     then ?leave
  loop ;

: APPEND    ( -- )  \ allonge le fichier
  begin append-blank false
        begin drop []edit retry? until
  until ;

eof  \ FIN DE FICHIER

*   dBCOOK : IMPORTATION ET EXPORTATION DE DONNEES AU  *
          *   STANDARD dBASE (c)  *
```

dBCOOK est un utilitaire Turbo-Forth de gestion de fichiers de données au standard bien connu dBASE II-III-III+ . Il permet notamment de consulter un fichier dBASE, de le modifier, d'y ajouter des données. Avec ses 4K compilés, dBCOOK représente un noyau interface entre un langage compilé puissant et un gestionnaire de bases de données universel. A partir de ce noyau simple, l'utilisateur pourra développer en Forth des applications compilées très spécifiques à son environnement dBASE personnel.

Bien que dBASE soit particulièrement puissant, flexible, aisément programmable grâce à son langage propre, il peut s'avérer nécessaire de disposer d'outils pour lesquels il n'a pas été prévu ou pour lesquels il peut sembler mal adapté ou trop lourd. Il est en outre appréciable de pouvoir partager à faible coût des fichiers dBASE sur des sites ne disposant pas du programme gestionnaire dBASE...

dBCOOK n'est pas un clone de dBASE: il se contente de gérer en syntaxe Forth les fichiers .DBF qu'utilise dBASE. Ses limitations au niveau de cette première version doivent être connues:

- ouverture, lecture et écriture d'un seul fichier .DBF à la fois
- seuls les fichiers .DBF sont traités.
- les champs libres ou dits "mémo" qui sont stockés à part par dBASE dans des fichiers .DBT ne sont pas traités.
- les variables dBASE (fichiers .MEM) ne sont pas échangeables sauf par le biais de fichiers .DBF particuliers.
- l'indexation (fichiers .NDX) n'est pas utilisée pas plus que les divers fichiers de format d'écrans, d'états, d'étiquettes ou les filtres d'extraction de données propres à dBASE.
- un fichier .DBF pour une bonne compatibilité doit être créé par dBASE lui-même, vide au besoin.
- la structure d'un fichier .DBF n'est pas modifiable.

Pour illustrer l'emploi de dBCOOK sous Turbo-FORTH, nous prendrons l'exemple d'un fichier ANNUAIRE.DBF comportant pour chaque enregistrement les champs NOM, PRENOM, ADRESSE, CODEPOSTAL, TELEPHONE.

1) Ouverture du fichier : USE ANNUAIRE

L'extension .dbf est facultative. Le mot marqueur SESSION est créé dans le vocabulaire DATABASE. Il est suivi de la création automatique des mots NOM PRENOM ADRESSE CODEPOSTAL et TELEPHONE qui permettent d'invoquer simplement les champs du fichier.

2) Affichage de la structure : .STRUCTURE

Ce mot équivaut au 'DISPLAY STRUCTURE' de dB qui renseigne sur le fichier. On a ajouté la taille du fichier et la taille de l'en-tête de fichier pour permettre le calcul de l'occupation du fichier:

```
taille totale  = longueur d'en-tête
 + ( nombre d'enregistrements
     * longueur d'enregistrement )
 + 1
```

3) Consultation du fichier : DISPLAY

En début de session, tous les champs sont actifs et aucune condition de sélection n'est imposée: DISPLAY affiche alors tout le fichier. L'affichage peut être suspendu ou avorté (STOP? ).

L'option RECNOS OFF supprime l'affichage des numéros d'enregistrements. RECNOS ON le rétablit. Il est en outre possible de modifier la largeur d'impression dans la variable COLS initialisée à 80 colonnes pour l'écran.

DISPLAY peut surtout être préalablement paramétré pour n'afficher que certains champs issus de certains enregistrements répondant à une condition quelconque.

4) Pointer un enregistrement : GO

3 GO charge dans le tampon l'enregistrement n°3. Dès lors, l'enregistrement courant donné par RECORD est 3 et toute opération d'entrée-sortie de données s'effectuera sur cet enregistrement.

1 GO et CAPACITY GO pointent le premier et le dernier enregistrement, CAPACITY délivrant le nombre total d'enregistrements.

Par convention, les mots agissant sur l'enregistrement courant débutent par '[]' :

```
4 GO []DISPLAY  affiche l'enregistrement 4
  7 GO []EDIT     édite l'enregistrement 7
```

L'ordre des enregistrements dans un fichier peut être modifié par COPY et PERMUTE qui copie ou permute l'enregistrement courant dans un autre enregistrement qui devient lui-même courant:

```
1 GO 5 COPY copie l'enregistrement n°1 dans le n°5
2 GO 7 PERMUTE permute les enregistrements n°2  et n°
7
```

COPY et PERMUTE ouvrent la possibilité de trier le fichier.

5) Extraire une donnée : ()

Toutes les données DBF sont en chaines ascii: () délivre pour Forth la chaine explicite ( --adr,len) du champ courant de l'enregistrement courant. Un champ devient courant par l'exécution de son nom compilé dans le vocabulaire à l'ouverture du fichier:

```
10 GO PRENOM () TYPE
   affiche le prénom du 10ème record
CODEPOSTAL ()  2 LEFT$  TYPE
   affiche les deux premiers chiffres du
   code postal de ce même enregistrement.
```

Par convention encore, les mots agissant sur le champ courant débutent par '()':

```
PRENOM ()DISPLAY
CODEPOSTAL ()OVAL D.
```

6) Introduire une donnée : REPLACE

Une chaine explicite Forth (adr,len--) est placée dans le champ courant de l'enregistrement courant.

```
" MARCEL" PRENOM REPLACE
CODEPOSTAL " 75001" REPLACE
```

REPLACE effectue un minimum de contrôles et formatage pour ne pas endommager le fichier avec des données invalides. Les champs numériques doivent disposer du point décimal en bonne position, les champs dates doivent être sous la forme de stockage DBF c'est-à-dire " 19841225" pour 25/12/84 (format AAAAMMJJ pour JJ/MM/AA). Le remplissage des blancs à droite pour les champs alpha, à gauche pour les champs numériques est assuré par REPLACE. Si une chaîne invalide est utilisée par REPLACE, le champ reste blanc sans message d'erreur: libre à l'utilisateur de filtrer, contrôler et valider ses données d'où qu'elles proviennent.

() et REPLACE forment donc la base de l'interface import-export entre Forth et un fichier .DBF

7) Sélection des champs actifs:
SELECTION SET FIELDS EXCEPT

Pour les procédures répétitives, certains champs peuvent être sélectionnés comme actifs. SELECTION prépare une sélection en rendant tous les champs inactifs. SET active le champ courant. FIELDS active tous les champs. EXCEPT désactive le champ courant.

```
SELECTION NOM SET TELEPHONE SET DISPLAY
  affiche le fichier avec
 les noms et téléphones comme seuls champs.

FIELDS ADRESSE EXCEPT DISPLAY
  affiche le fichier sans les adresses
```

Une sélection de champs reste active pour toutes les procédures ultérieures jusqu'à une nouvelle sélection (FIELDS par exemple).

8) Condition dans une procédure : FOR

Les procédures répétitives ne sont exécutées que si le vecteur CONDITION délivre un drapeau vrai. FOR vectorise CONDITION sur un mot Forth délivrant une condition pour chaque enregistrement.

```
 : DIX-PREMIERS RECORD 10 <= ;
FOR DIX-PREMIERS DISPLAY
  affiche les dix premiers enregistrements

 : TEST PRENOM " MICHEL" ()= ;
FOR TEST DISPLAY
  affiche les enregistrements où prénom='MICHEL'
```

Les mots ()= ()>= ()<= ()$ ajoutent des tests simples portant sur le champ courant aux opérateurs chaînes usuels du Forth. Les conditions sont mixables entre elles et avec les sélections de champs:

```
 : PARIS CODEPOSTAL () 2 LEFT$ " 75" COMPARE$ 0= ;
 : *DUR* " DUR" NOM ()$ ;
 : TEST2 PARIS *DUR* AND ;
RECNOS OFF SELECTION TELEPHONE SET
FOR TEST2
DISPLAY
  affiche les numéros de téléphone
  des noms contenant 'DUR' et habitant Paris,
  sans les numéros d'enregistrements.
```

Une condition reste active pour toutes les procédures ultérieures. Le mot ALL supprime toute condition précédente: tous les records sont alors pris en compte.

9) Effacement d'enregistrements : DELETE

Cet effacement est une procédure conditionnée par FOR qui ne fait que marquer comme dBASE les enregistrements à détruire. Ils sont repérés par le signe '*' en tête d'enregistrement. Il faut utiliser la commande PACK de dBASE pour réorganiser le fichier sans les enregistrements supprimés.

```
FOR PARIS DELETE
  efface les parisiens de l'annuaire.
```

[]DELETE efface l'enregistrement courant:

```
3 GO []DELETE
```

[]RECALL le restitue et RECALL est la procédure de restitution sur le fichier. DELETED et UNDELETED fournissent les conditions 'détruit' et 'valide' pour l'enregistrement courant.

```
 : TEST3 DELETED DIX-PREMIERS NOT AND ;
FOR TEST3 RECALL
  restitue les parisiens précédemment effacés s'ils
  ne sont pas dans les 10 premiers enregistrements.
```

10) Editions d'enregistrements : EDIT

Cette procédure travaille également en sélection et condition pour entrer des données au clavier. Chaque édition d'enregistrement peut être reprise avant de passer à la suivante.

```
FOR DIX-PREMIERS EDIT
  demande l'édition des 10 premières fiches.
```

Les mots ()EDIT et []EDIT éditent un champ ou un enregistrement :

```
3 GO []EDIT
5 GO TELEPHONE ()EDIT
```

11) Ajouts d'enregistrements : APPEND

Ce mot entre en édition de nouveaux enregistrements placés en fin de fichier. APPEND-BLANK est la primitive d'ajout d'un enregistrement vide.

12) Procédures vectorisées : PROCEED

La puissance Forth permet d'envisager des actions complexes sur le fichier. Supposons que le mot LETTRE soit défini pour imprimer une lettre personnalisée à partir des champs NOM, ADRESSE et CODEPOSTAL d'un enregistrement. En vectorisant le différé PROCESS, le mot PROCEED appliquera cette action sur tout le fichier en fonction de la condition en cours:

```
FOR PARIS PROCEED LETTRE
  imprime une lettre personnalisée pour
  chaque parisien.
```

Le mot .COUNT permet d'afficher le nombre d'enregistrements sur lesquels la dernière procédure a été effectuée.

13) Fermeture du fichier .DBF : END

Ce mot ferme le fichier et restitue le dictionnaire en effaçant les mots créés lors de la session de travail.

En cas d'erreur Forth, le fichier est automatiquement refermé. Il convient alors d'effectuer un FORGET SESSION ou un END (qui peut aussi émettre une erreur sans importance) pour ne pas surcharger le dictionnaire avec des noms de champs recréés à chaque USE.

dBCOOK permet donc la consultation et la maintenance minimale d'un fichier dBASE. Il autorise surtout le développement d'applications compilées spécifiques sur cette base de données.

---

COURRIER: DEBUTANT

Etant débutant en FORTH, je me heurte à quelques problèmes. Aussi je me permets de contacter l'association JEDI à laquelle j'ai adhéré depuis peu.
Celà fait un mois que je me suis lancé dans ce langage, d'abord sur ATMOS et maintenant compatible PC.
Là, toute la mémoire n'étant pas adressable en deux octets, je ne trouve pas l'ordre permettant de sortir de la zone du FORTH (pour accéder à la mémoire écran par exemple).
D'autre part, comment procéder pour appeler à partir du BASIC un sous-programme écrit en FORTH. Voilà les deux problèmes qui gênent mon sommeil!

Didier BRANDA - 75015 PARIS

REPONSE:
Les ordres permettant d'accéder à toute la mémoire sont LC!, LC@, L! et L@ (équivalents de C!, C@, ! et @) et admettent comme paramètres:

```
c seg off  LC!
  seg off  LC@  --- c
n seg off  L!
  seg off  L@   --- n
```

où seg et off sont les valeurs correspondant au segment mémoire et le décalage dans ce segment. Exemple, l'adresse 0:16 (segment 0, décalage 16) est équivalente à 1:0 (segment 1, décalage 0), un segment étant un multiple de 16 octets par rapport à l'adressage absolu; le décalage est situé dans l'intervalle 0..65535.
Quant à l'exécution d'un sous-programme FORTH depuis BASIC, nous ne savons pas... mais est-ce vraiment utile? Avec de l'expérience en FORTH, vous vous passerez très bien de BASIC.

---

COURRIER: EN VRAC

Voici en vrac quelques questions qui font patiner quelque peu mon démarrage en FORTH:
* comment décompresser un programme (META80.BQK) sur AMSTRAD, quand l'étendue ne tient pas sur une disquette?
* j'ai appris que RAPID-FILE pouvait récupérer des fichiers d'autres applications, développées sous dBASE entre autres (question: quel est ce format?). Alors comment accéder à un fichier sous MSDOS ou TOS à la manière d'un PASCAL ou d'un C: caractère par caractère jusqu'à EOF. J'imagine qu'on peut récupérer le pointeur sur un bloc,

parcourir le bloc, passer au suivant, et ainsi de suite. Mais je n'ai pas trouvé le pointeur, je ne sais pas si c'est la seule solution, ni la plus élégante. Alors comment faire?

* pourrai-je avoir la carte mémoire du VolksFORTH? Où sont les registres qui contiennent IP, SP, RP?.

Christophe BELLE - 38400 ST MARTIN D'HERES

REPONSE:
A la première, une seule solution, décompresser META80.BQK sur un PCW avec disque virtuel>180K, couper le fichier en deux et le remettre sur disquette en deux parties. Pour les détails, se reporter au manuel CP/M. Mais peut-être un autre adhérent y est-il arrivé et peut-être vous portera secours: appel aux bonnes volontés.
Concernant RAPID-FILE, je ne l'ai guère manipulé et ne peut donc confirmer. Si quelqu'un veut nous faire un exposé sur les manips vicieuses de RAPID-FILE. Quand à la manipulation de fichiers et de lecture caractère par caractère, peut-être avez-vous trouvé réponse à cette question dans des récents articles de JEDI: fichier UFILES.FTH et DBCOOK.FTH. Enfin, concernant la carte mémoire de VolksFORTH, si quelqu'un l'a, il nous la communiquera.

---

COURRIER: ARBRES B

Je viens de recevoir différents ouvrages américains sur le FORTH, notamment les FORML PROCEEDINGS 1985 et 1986, et dans ces deux ouvrages, il y a un article de Mr Martin J. TRACY sur un micro LISP en FORTH, et il fait référence à un article qu'il a publié dans le ROCHESTER 1986. Si quelqu'un pouvait me faire parvenir la photocopie intégrale de cet article, il serait bien aimable de me le faire parvenir à mon adresse. Je l'en remercie d'avance.
Je cherche quelqu'un qui pourrait m'expliquer clairement les arbres B pour la gestion des fichiers indexés, et j'aimerais connaître les avantages et inconvénients de ce système par rapport à l'organisation ISAM.
Si des exemples en C ou en FORTH pouvaient m'être fournis, celà serait l'idéal.
Egalement, je suis actuellement en train de développer une gestion de fichier (BIBLE) de textes comprenant des formules pré-établies pour des juristes (notaires, avocats, et autres...). Pour faciliter la liaison dans mes fichiers de textes types, j'ai eu l'idée de coder la gestion de la liste précédente ou suivante de la manière ci-après:

- fichier codé sur 128 octets, 80 pour le texte, le reste pour le nom du paragraphe, la ligne de paragraphe, le type de texte, et ensuite quelques octets pour des développements ultérieurs, et à l'intérieur, j'ai un chainage sur l'enregistrement précédent et suivant.
- si l'enregistrement en qusetion a, pour exemple:
  pour précédent: 28562
  pour suivant : 52456

je code sur 5 octets les deux adresses en les mixant sous forme hexadécimale 25 82 54 65 26. Mais j'ai des problèmes pour faire la conversion dans ce sens et dans le sens contraire, il est ici précisé qu'il n'y a pas de lettre A, B, C, D, E et F dans ces adresses, le dernier enregistrement étant signalé par FFFFF.
Je désirerai que quelqu'un puisse m'apporter ses lumières pour pouvoir manipuler les octets de façon à ce que je puisse pouvoir opérer sur 1 à dix octets de long, notamment je désire porter cette utilisation en FORTH et C.

Jean-Marc COURET - 9, rue Racine - 83000 TOULON

---

COURRIER: OUTRE-RHIN

(extrait:) UPC wirkt nicht auf dir deutschen Buchstaben ä, ö und ü. Wir können aber im Deutschen auf diese Buchstaben nicht verzichten. Außerdem ist nicht einzusehen, was dabei zu verlieren wäre, wenn man sie mit in UPC einbeziehen würde. Daß dir Amerikaner ihre Existenz ignorieren. Forth ist für alle da. Also auch für uns. Ich plädiere für eine Einbezienung.

Fred BEHRINGER - D-8000 MÜNCHEN 40

TRADUCTION (Versetzung):
UPC ne traite pas les caractères allemands ä, ö et ü. Nous ne pouvons pourtant pas nous passer d'eux. Il serait pourtant utile que ces caractères puissent être convertis par UPC. Soit, les américains ignorent leur existence. Forth est fait pour tout le monde. Aussi pour nous. Je plaide pour une modification.

REPONSE:
Il est vrai que ni F83, ni TURBO-Forth ne traitent les caractères accentués d'origine étrangère. Et dieu sait s'il y en a encore plus en français qu'en allemand. La modification de UPC est somme toute mineure et je profite de votre courrier pour lancer un challenge: définir un nouveau UPC tenant compte de votre suggestion. A convertir:

```
      à â é è ê î ï ô ù û ç   pour les français
  en  A A E E E I I O U U C
  ou      É               Ç
      ä ö ü                   pour les allemands
  en  A O U
  ou  Ä Ö Ü
```

(les deux options pour les pratiquants du patois luxembourgeois écrit...), mais aussi

```
      å                       pour les norvégiens
  en  A
  ou  Å
      ñ                       pour les espagnols
  en  N
  ou  Ñ
```

---

FORTH

## COMPILATEUR BILINGUE: FORTH 79 et 83

Par Mr KERJEAN

Pour que tout le monde puisse parler le même langage, le FORTH 83, voici sa définition, à partir du FORTH 79.

A ceux qui veulent conserver des programmes en FORTH 79, il suffit de mettre les nouvelles définitions dans un vocabulaire particulier en écrivant:

VOCABULARY FORTH-83

DEFINITIONS SUPPLEMENTAIRES:
Pour rendre ces définitions plus générales, plus lisibles et plus compactes, il a été défini trois types de définitions: des constantes du compilateur, des abréviations, et des définitions courament employées. Les voici:

1) constantes du compilateur:
   +D longueur d'une cellule, en octets,
   -S incrément d'adresse pour passer d'une cellule de la pile de données à la précédente,
   -R idem, pour la pile de retour.

2) abréviations:

| abrév. | expressions |
|---|---|
| ;I | ; IMMEDIATE |
| [] | [COMPILE] |
| ][ | LITERAL |
| ?( | IF |
| () | ELSE |
| )- | THEN |
| ^ | DUP |
| V | OVER |
| \| | SWAP |
| !! | DROP |

La parenthèse gagne à être remplacée par l'accolade, qui ne figure généralement pas sur les machines à écrire; en mise au point de programme, il est souvent commode d'invalider provisoirement du code en le mettant entre parenthèse; il vaut donc mieux minimiser les ) .

3) mots supplémentaires

Il s'agit de CASE, OF, ENDOF, ENDCASE ainsi que ASCII, d'usage courant et qui seront utilisés par la suite.

CASE dépose un simple marqueur.
OF et ENDOF sont équivalents à IF et ELSE.
ENDCASE résout les références aux différents ENDOF jusqu'à la rencontre du marqueur déposé par CASE.
ASCII dépose le code ASCII du premier caractère du mot qui le suit; en compilation, il est compilé comme un litéral.

## DEMARCHE SUIVIE

A partir des documents existants:

- définition du FORTH 83,
- définition du FORTH 79,
- articles déjà parus, indiquant comment réécrire des programmes écrits en FORTH 79.

Les définitions faciles ont été écrites, puis celles moins faciles: il ne restait plus alors que deux:

WORD dont la définition faisait appel à ENCLOSE, défini en assembleur; son action exacte a été retrouvée à partir de l'état de la pile avant et après ENCLOSE.

LEAVE; sa définition a longtemps paru impossible à écrire sans une refonte complète de DO et LOOP. La solution a été trouvée en écrivant, sur papier, l'équivalent exact: il suffit de faire suivre LEAVE de 0 IF, et de résoudre par une série de THEN, les références à IF. Le mot LEAVE a donc été créé, qui:

- dépose 0 sur la pile,
- nettoie la pile retour, avant d'appeler
- LEAVE qui rend égaux les deux indices.

LEAVE), incorporé à LOOP et +LOOP, résout les références à LEAVE.

Le mot IF dépose, outre l'adresse de destination, un marqueur, spécifique du compilateur. Pour le récupérer, il aurait été possible de le faire par: DUP ROT ROT.

Il est plus court de passer par al pile de retour, la lecture se fait au cours de la définition de IN1@.

LEAVE dépose le marqueur de IF, modifié, et LEAVE) le rétablit, avant d'appeler la partie exécutive de THEN.

Quand, au cours d'une définition, on écrit dans la pile de retour, il faut être sûr qu'elle soit bien rétablie avant la fin du flot d'entrée, BUFFER, ou tampon de clavier... sous peine de surprise!!!

## DIFFICULTES RENCONTREES

Outre celles, prévues, déjà mentionnées, d'autres ont été rencontrées tout à fait ailleurs que prévu:

- R@ le mot : R@ R ; dépose simplement son adresse de retour! sa définition aurait pu être : R@ R R) SWAP )R ;

Comme il fallait définir I', il est plus court d'écrire:

: I' J ; : R@ I' ;

- ['] et ' leur définition paraissait évidente, mais l'exécution d'un tel MOT était souvent "surprenante"; ' avait donc été défini à partir de -FIND; le problème n'a été bien résolu qu'après avoir vérifié toutes les définitions, dont LITERAL, qui, en FORTH 79 ne s'exécute qu'en compilation.

- LEAVE c'est en fait un branchement vers le premier LOOP ou +LOOP rencontré. L'adresse de résolution peut difficilement être mise sur la pile, car ce branchement s'enchevêtre avec les imbrications IF, BEGIN et DO.

Elle a été mise en pile RETOUR, mais la mise au point du mot a été difficile; tous les moyens de mise au point ont été utilisés, dont certains seront exposés par la suite, c'est à dire:

- interprétation des messages d'erreur,
- impression des piles données et retour
- décompilation,
- compilation en adresse fixe.

Cela n'a pas suffi, il a fallu en utiliser un autre, l'heuristique, c'est à dire les essais et erreurs qui est très déconseillée par ailleurs, mais parfois indispensable. néanmoins, après une journée de mise au point, émaillée de nombreux rechargements, le mot LEAVE était au point.

## CONCLUSION

Le langage FORTH est un langage aux possibilités étonnantes: une mise à jour du dictionnaire, portant sur 48 mots, a pu être écrite en 32 lignes de 64 caractères, en n'utilisant que les ressources propres du langage!

LISTING:

```
: ;I [COMPILE] ; IMMEDIATE : IMMEDIATE
: LITERAL COMPILE LIT , ;I
: [] [COMPILE] [COMPILE] ;I
: ]L LITERAL ;I
: VARIABLE 0 VARIABLE ;
: ^ DUP ;
: ¦ SWAP ;
TIB @ CONSTANT TIB
VARIABLE SPAN
: V OVER ;
: !! DROP ;
0 2 ^ CONSTANT +D
VARIABLE #TIB
: NOT -1 XOR ;
: 1- 1 - ;
- CONSTANT -D
: = = MINUS ;
: 0= 0 = ;
: UM* U* ;
: 2- 2 - ;
+D CONSTANT -S
: ) ) MINUS ;
: 0) 0 ) ;
: 2* DUP + ;
: 2/ 2 / ;
+D CONSTANT -R
: ( ( MINUS ;
: 0( 0 ( ;
: UM/MOD U/ ;
: I' J ;
: R@ I' ;
: U( U( MINUS ;
: U) ¦ U( ;
: NEGATE MINUS ;
: DNEGATE DMINUS ;
: )BODY +D + ;
: )LINK -D + ;
: )NAME )BODY NFA ;
: ?{ [] IF ;I
: BODY) CFA ;
: LINK) +D + ;
: NAME) PFA CFA ;
: {} [] ELSE ;I
: CREATE (BUILDS ;
: DEPTH SP@ S0 @ ¦ - -S / ;
: }- [] THEN ;I
: PICK 1+ -S * SP@ + @ ;
: ROLL )R R PICK SP@ R -S * + R) 1+ 0
  DO ^ @ V -S + ! -S - LOOP !! !! ;
: ABORT S0 @ SP! QUIT ;
: ST! STATE @ )R STATE ! EXECUTE R) STATE ! ;
: ?DUP -DUP ;
: ['] [ ' ' CFA ]L -1 ST!
  -D HERE V + +! ;I
: )IN IN ;
: COMPILE R) ^ +D + )R @ , ;
: ' ['] ' 0 ST! -D + ;
: (ABORT")
  ?{ R COUNT TYPE ABORT
  {} R) COUNT + )R  }- ;
: ABORT" COMPILE (ABORT")
  34 WORD HERE C@ 1+ ALLOT  ;I
: EXPECT V V EXPECT !! )R R 1-
  BEGIN 1+ ^ C@ 0=
  UNTIL R) !! ;
: .(  29 WORD HERE COUNT TYPE ;I
: EXIT COMPILE ;S ;I
: WORD ¦ HERE )R DP ! WORD R) DP ! ;
: SAVE-BUFFERS FLUSH ;
: IN1@ BLK @
   ?{ [ )R R ] BLK @ BLOCK
   {} TIB }- ;
```

```
: FIND  >R R IN @ >R IN1@ - in ! -FIND R> IN !
  ?( R> !! >R CFA R> 64 AND -32 / 1+ () R> 0 )- ;
: DO R> 108 >R >R [] DO ;I
: (LEAVE) R> LEAVE >R 0 ;
: LEAVE COMPILE (LEAVE) R> [] IF ; >R >R >R ;I
: LEAVE> R> R> BEGIN I [ R ]L =
  WHILE R> R> ; [] THEN REPEAT R> 108 - ABORT" LEAVE?"
  >R >R ;
: LOOP LEAVE> [] LOOP ;I
: +LOOP LEAVE> [] +LOOP ;I
: CMOVE> >R V - >R 1- ^ I' + DO I C@ I J + C! -1 +LOOP
  R> DROP R> DROP ; R>
```

FORTH

# PROGRAMMATION EN LANGAGE ORIENTE OBJET

par Terry RAYBURN
adaptation Marc PETREMANN

Systèmes: tous systèmes F83

On parle depuis peu de nouveaux langages, ceux de cinquième génération travaillant avec des objets, tels SMALLTALK ou NEON. Et si FORTH leur a donné de l'inspiration, ils peuvent contribuer à faire évoluer FORTH.

## UNE APPROCHE DES LANGAGES ORIENTES OBJET

Le concept de langage orienté objet semble à priori très confus, mais en l'approfondissant, on peut le greffer sans difficulté en FORTH. C'est à partir de deux articles, le premier "MULTI-DOES>" de A.GOPPOLD, le second "METHODS> OBJECT-ORIENTED EXTENSIONS REDUX" de Terry Rayburn, parus dans Vierte DIMENSION (qui est en RFA ce que JEDI est en France...), que cette idée s'est imposée.

Tout ceux pratiquant FORTH savent que l'on traite des données de différentes nature:

```
  données 16 bits (signées ou non)
  données 32 bits (    "        " )
  octets
  chaînes de caractères
etc...
```

liste pouvant être étendue à des vecteurs, matrices, bits, tableaux multi-dimensionnels, etc...

Pour illustrer le propos, prenons les trois premières catégories et les trois actions les plus fréquemment employées:

| | stockage | empilage | affichage |
|---|---|---|---|
| octet | C! | C@ | C? si défini |
| 16 bits | ! | @ | ? |
| 32 bits | 2! | 2@ | 2? si défini |

L'opération de "stockage" peut être décrite génériquement par un mot quelconque PUT, STORE, AFFECTE... et pouvant s'appliquer à un type quelconque de données:

```
  octet adresse   PUT  (au lieu de C!)
  16-bits adresse PUT  (au lieu de !)
  32-bits adresse PUT  (au lieu de 2!)
```

ce qui réduirait considérablement les opérateurs dfe manipulation de données, les mots C!, ! et 2! devenant des primitives.

Pour arriver à ce résultat, on affecte un numéro d'ordre à chacune des actions prévues sur les objets considérés:

```
  action 1 pour GET en remplacement de  C@  @  2@
  action 2 pour PUT en remplacement de  C!  !  2@
  action 3 pour VUE en remplacement de  C?  ?  2?
    (sous réserve de définition de C? et 2?)
```

et on définit chaque type de données, par une séquence de la forme:

```
  : HALF     CREATE 0 C,    DOES> EXIT C@ C! C? ;
  : INTEGER  CREATE 0 ,     DOES> EXIT  @  !  ? ;
  : DOUBLE   CREATE 0 , 0 , DOES> EXIT 2@ 2! 2? ;
```

Bien entendu, il faudra discriminer les types 16 bits signés et 16 bits non signés; idem pour les 32 bits. Je vous laisse ce soin.

L'apparition du mot EXIT placé systématiquement après DOES> revient à définir un nouveau type de mot de définition. Reprenons la syntaxe définie par Terry Rayburn:

```
  METHOD: définit une action utilisée en interprétation
  [METHOD]: définit une action utilisée en compilation
seulement
  : <mot-classe>
  CREATE  <paramètres>
  METHODS> <série-d'actions> ;
```

et dont les définitions sont les suivantes:

```
  \ Cette partie est valable pour tout système F83
  : METHODS>
     [COMPILE] DOES>  COMPILE EXIT ; IMMEDIATE
  : [METHOD]:
     CREATE 2* , IMMEDIATE
     DOES>  HERE 2- @ @
     5 + SWAP @ + @ , ;
  : METHOD:
     CREATE 2* ,
     DOES> @ OVER 2- @ 5 + + PERFORM ;
```

On définit les types d'actions utilisables en interprétation et en compilation:

```
  0 METHOD: GET        1 METHOD: PUT        2 METHOD: VUE
  0 [METHOD]: [GET]      1 [METHOD]: [PUT]   2 [METHOD]:
[VUE]
```

Exemple:

```
  : C? C@ . ;
  : HALF
     CREATE 0 C,
     METHODS> C@ C! C? ;
  HALF CARACTERE
  7 CARACTERE PUT  \ affecte 7 à l'objet CARACTERE
    CARACTERE GET  \ empile le contenu de l'objet
    CARACTERE VUE  \ affiche le contenu de l'objet
```

et en compilation:

```
  : <mot> ....   7 CARACTERE [PUT] ... ;
```

Le transfert du contenu d'un objet 8 bits vers un objet 16 bits sera de la forme:

```
  : INTEGER
     CREATE 0 ,
     METHODS> @  !  ?  ;
  INTEGER DONNEE
  CARACTERE GET DONNEE PUT
```

et mieux, mélanger les opérations sur les données:

```
  CARACTERE GET
  DONNEE GET + DONNEE PUT
```

à condition de redéfinir les opérateurs pour qu'ils traitent invariablement des données de type différent.

## APPLICATION A LA GESTION DES PORTS SERIE

Systèmes: spécifique F83 MSDOS Laxen et Perry ou Turbo-Forth.

Pour illustrer ce qui me semble un concept nouveau, aussi nouveau que l'a été $EXECUTE ou ONLY..ALSO, on a définit toutes les fonctions de gestion du port de communication.

LISTING:

```
\ A partir d'ici, le programme concerne exclusivement
\ les materiels IBM PC,XT,AT et compatibles
HEX
VARIABLE PORT.ADRESS
: COM1   \ affecte l'adresse du port série 1
   3F8 PORT.ADRESS ! ;
: COM2   \ affecte l'adresse du port série 2
   2F8 PORT.ADRESS ! ;

: +P ( a --- adr) \ délivre adresse port serie + offset
   PORT.ADRESS @ + ;

\ décalage de bits
CODE <SH ( n m --- n')
   CX POP   AX POP
   BEGIN  CX ROR   8000 # CX TEST  0=
   WHILE  AX SHL
   REPEAT  1PUSH   END-CODE
CODE >SH ( n m --- n')
   CX POP   AX POP
   BEGIN  CX ROR   8000 # CX TEST  0=
   WHILE  AX SHR
   REPEAT  1PUSH   END-CODE
DECIMAL

\ Type de données 8 bits
: @BITS  ( a --- n)
   2@ +P  PC@ OVER AND SWAP >SH ;
: !BITS  ( a n ---)
   2@ 2DUP +P PC@ SWAP -1 XOR AND >R
   >R DUP >R <SH R> AND  R> SWAP R> OR SWAP +P PC! ;
: .BITS ( a ---)
   @BITS . ;
: BITS: ( m n ---)
   CREATE , ,
   METHODS> @BITS !BITS .BITS ;

\ Registres de validation d'interruption
1 1 BITS: EDAI        \ donnée reçue et prête
2 1 BITS: ETXHREI     \ registre d'attente
                      \ de transmission vide
4 1 BITS: ERLSI       \ caractère reçu avec condition
                      \ d'arrêt ou d'erreur
8 1 BITS: EMSI        \ modification dans l'état du modem

\ Registres d'identification d'interruption
1 2 BITS: -IP          \ 0  : condition d'interruption
                           \ en attente
                       \ 1  : aucune interruption en attente
6 2 BITS: IID          \ D1 et D2  : identification des
                           \ interruptions

\ Registres de contrôle de ligne
  3 3 BITS: WLS        \ bits 0 et 1, longueur du caractère
                       \     0 0  : 5 bits
                       \     0 1  : 6 bits
                       \     1 0  : 7 bits
                       \     1 1  : 8 bits
  4 3 BITS: STB        \ bits d'arrêt,
                       \     0  : 1 bit d'arrêt
                       \     1  : 1,5 bit si longueur
                           \ caractère = 5 bits
                       \     2       si longueur
                           \ caractere = 6, 7 ou 8 bits
  8 3 BITS: PEN        \ bit de parité
                       \     0  : aucun bit de parité
                       \     1  : parité imposée
 16 3 BITS: EPS        \ type de parité
                       \     0  : parité impaire
                       \     1  : parité paire
 32 3 BITS: STICK.PARITY  \ blocage de parité
                       \     0  : hors service
                       \     1  : si les bits 3 et 4 valent 1
                           \ parité toujours 0
                       \       si b3 = 1 et b4 = 0, parité
                           \ toujours 1
 64 3 BITS: SET.BREAK  \ générateur de "break"
                       \     0  : hors service
                       \     1  : sortie UART forcée au 0
                           \ logique
128 3 BITS: DLAB       \ DLAB,  pour accès au diviseur de
                           \ frequence
```

```
\ Registres état de ligne
  1 5 BITS: DR         \ un caractère est disponible
  2 5 BITS: ORE        \ erreur d'engorgement
  4 5 BITS: PE         \ erreur de parité
  8 5 BITS: FE         \ erreur d'encadrement (bit d'arrêt)
 16 5 BITS: BI         \ détection de "break"
 32 5 BITS: THRE       \ registre d'attente de
                           \ transmission vide
 64 5 BITS: TSRE       \ registre de décalage de
                           \ transmission vide

\ Registres de contrôle de modem
  1 4 BITS: DTR        \ ordinateur prêt
  2 4 BITS: RTS        \ demande pour émettre
  4 4 BITS: OUT1       \ OUT 1, sortie 1
  8 4 BITS: OUT2       \ OUT 2, sortie 2
 16 4 BITS: LOOPBACK   \ si ce bit est mis à 1, la sortie
                       \ série est envoyée
                       \ sur l'entrée. Sert pour autotest.

\ Registres état du modem
  1 6 BITS: DCTS       \ "DELTA" prêt à émettre
  2 6 BITS: DDSR       \ "DELTA" modem prêt
  4 6 BITS: TERI       \ "DELTA" détection de sonnerie
  8 6 BITS: DRLSD      \ "DELTA" connexion établie
 16 6 BITS: CTS        \ prêt à émettre
 32 6 BITS: DSR        \ modem prêt
 64 6 BITS: RI         \ détection de sonnerie
128 6 BITS: RLSD       \ connection établie

\ Données en lecture seule
: @ROR  ( a --- n)
   @ +P PC@ ;
: -DEFINED
   ABORT" METHOD non définie pour ce type de données " ;
: .ROR ( a ---)
   @ROR EMIT ;
: READ.ONLY: ( n ---)
   CREATE ,
   METHODS>  @ROR -DEFINED .ROR ;

\ Données en écriture seule
: !WOR ( n a ---)
   2DUP ! 2+ @ +P PC! ;
: .WOR ( a ---)
   @ EMIT ;
: WRITE.ONLY: ( n ---)
   CREATE 0 , ,
   METHODS> @ !WOR .WOR ;

\ Données spéciales
: @SR  ( a --- n)
   1 DLAB [PUT] @ +P P@ 0 DLAB [PUT] ;
: !SR ( n a ---)
   1 DLAB [PUT] @ +P P! 0 DLAB [PUT] ;
: .SR ( a ---)
   @SR . ;
: SPECIAL: ( n ---)
   CREATE ,
   METHODS> @SR !SR .SR ;

\ Registres restants
0 READ.ONLY: RX          \ registre de réception de données
0 WRITE.ONLY: TX         \ regsitre de transmission de données
0 SPECIAL: DIVISOR.LATCH  \ diviseur de fréquence pour
           \ contrôle de la vitesse de transmission

\ Initialisation du 8250
: BAUD ( n ---)
   >R 1843200. 16 MU/MOD R> M/MOD
   DIVISOR.LATCH [PUT]  2DROP ;

3 CONSTANT EVEN        \ parité
0 CONSTANT NO
1 CONSTANT ODD
: PARITY ( n ---)
   2 /MOD EPS [PUT]  PEN [PUT] ;
: BITS ( n ---)
   5 - WLS [PUT] ;
: SET  ( n ---)
   TRUE SWAP   PUT ;
: RESET ( n ---)
```

```
   FALSE SWAP   PUT ;

\ Routines d'auto-test du 8250
: WAIT ( reg --- fl)
   32000 0
   DO    DUP GET
         IF    0= LEAVE   THEN
   LOOP
   DUP   IF    LOOPBACK RESET  THEN ;
: SELF.TEST ( ---)
   DIVISOR.LATCH [GET]  DUP   1+ DUP  DIVISOR.LATCH [PUT]
   DIVISOR.LATCH [GET] = NOT ABORT" port non valide "
   DIVISOR.LATCH [PUT]  LOOPBACK SET  0
   1024 0
   DO    I C@  DUP   TX [PUT]
         THRE WAIT  ABORT" Transmission défectueuse "
         TSRE WAIT
      ABORT" Registre décalage transmission défectueux "
         DR   WAIT  ABORT" Donnée non reçue "
         RX [GET] = NOT IF   1+   THEN
LOOP LOOPBACK RESET . ." /1024 erreurs de transmission " ;

\ Exemples d'initialisation du MINITEL <-> PC
\ COM1 1200 BAUD  7 BITS  EVEN PARITY  DTR SET  RTS RESET
\ CR .( COM1 auto-test en cours ) SELF.TEST

EOF \ FIN DE LISTING
```

La gestion du port de communication pouvait être réalisée de plusieurs manières:
- en code machine; maintenance et lisibilité difficile, mise au point laborieuse;
- par gestion des interruptions; en pratique, des aberrations de gestion du port série se produisaient et ne donnaient pas toujours le résultat escompté;
- entièrement en FORTH "classique"; prolifération des opérateurs et des primitives.

La solution adoptée ici consiste à reprendre la liste de TOUS les registres et les indicateurs du 8250 et à définir des classes d'objets pour chaque type de données:

- BITS: bits en lecture/écriture
- READ.ONLY: octets en lecture seule
- WRITE.ONLY: octets en écriture seule
- SPECIAL: adresse (diviseur de fréquence)

La lecture de DTR (DATA TERMINAL READY) est ainsi accessible très simplement par DTR VUE.

L'envoi d'un caractère vers le port de communication, après initialisation, est de la forme:

```
ASCII A TX PUT
```

INTERFACAGE AU MINITEL

Systèmes: F83 Laxen et Perry MSDOS et TURBO-Forth

Si vous disposez d'un câble de liaison MINITEL-PC (sinon je vous y encourage), voici des définitions permettant de gérer la communication entre le PC et le MINITEL.

LISTING:

```
: MINITEL ( ---)
   COM1
   1200 BAUD    \ communication à 1200 bauds
      7 BITS    \ 7 bits de données
   EVEN PARITY  \ parité
    DTR SET
    RTS RESET
   EDAI SET     \ indicateur donnée reçue et prête
   ;

: >RS ( c ---)
   TX [PUT]
   \ envoi caractère vers registre de transmission
   BEGIN  THRE [GET]
         \ attente registre de transmission vide
   UNTIL ;
HEX
VARIABLE ACCENTS   ACCENTS OFF
: (RSEMIT)  ( c ---)
   DUP 7F >
   IF                \ traitement des accentués
      ACCENTS @
      IF   \ transcodage en séquence car. accentués
        CASE
          ASCII à OF  19 >RS  41 >RS  61 >RS  ENDOF
          ASCII â OF  19 >RS  43 >RS  61 >RS  ENDOF
          ASCII é OF  19 >RS  42 >RS  65 >RS  ENDOF
          ASCII ê OF  19 >RS  43 >RS  65 >RS  ENDOF
          ASCII è OF  19 >RS  41 >RS  65 >RS  ENDOF
          ASCII î OF  19 >RS  43 >RS  69 >RS  ENDOF
          ASCII ô OF  19 >RS  43 >RS  6F >RS  ENDOF
          ASCII ù OF  19 >RS  41 >RS  75 >RS  ENDOF
          ASCII û OF  19 >RS  43 >RS  75 >RS  ENDOF
          ASCII ç OF  16 >RS  4B >RS  63 >RS  ENDOF
        ENDCASE
      ELSE   \ transcodage en car. non accentués
        CASE
          ASCII à OF  ASCII a >RS  ENDOF
          ASCII â OF  ASCII a >RS  ENDOF
          ASCII é OF  ASCII e >RS  ENDOF
          ASCII ê OF  ASCII e >RS  ENDOF
          ASCII è OF  ASCII e >RS  ENDOF
          ASCII î OF  ASCII i >RS  ENDOF
          ASCII ô OF  ASCII o >RS  ENDOF
          ASCII ù OF  ASCII u >RS  ENDOF
          ASCII û OF  ASCII u >RS  ENDOF
          ASCII ç OF  ASCII c >RS  ENDOF
        ENDCASE
      THEN
   ELSE
      >RS
   THEN ;
DECIMAL
: RSEMIT ( c ---)
   DUP   (EMIT)      \ envoi caractère vers console
   (RSEMIT)  ;       \ et également sortie série

: RSKEY? ( --- fl)
   DR [GET]  1 =     \ cherche si caractère reçu
   0 DR [PUT]  ;

: RSKEY ( --- c)
   BEGIN
      DR [GET] 1 =   \ on teste si donnée reçue
      (KEY?) OR      \ ou si appui sur touche clavier PC
   UNTIL
   RX [GET]          \ empile caractère reçu
   ;

: (RSCHAR)
   3DUP (EMIT) + C! 1+ ;

: LOGIN
  ['] (rschar) is char
  \ à n'utiliser qu'avec terminal local
  ['] rskey? is key?
  ['] rskey is key
  ['] rsemit is emit  ;

: LOGOUT
  ['] (char) is char
  ['] (key?) is key?
  ['] (key) is key
  ['] (emit) is emit  ;

\ version code machine de RSEMIT
\ CODE >RS  ( c ---)
\    PORT.ADRESS 5 + # dx mov  \ registre d'état de ligne
\    BEGIN
\       dx al in   20 # al test
\    0<> UNTIL
\    PORT.ADRESS # dx mov
\    ax pop            \ charge registre al avec octet
\    al dx out         \ et on l'envoye
\    NEXT END-CODE

\ FONCTIONS DE COMMUNICATIONS TELETEL

: (EMET") ( adl len ---) \  transmet  chaine  vers port
```

```
     LITERAL @                        \ prélève vecteur de EMIT
     ['] (RSEMIT) IS EMIT             \ remplace par (RSEMIT)
     R> COUNT 2DUP + >R TYPE
         \ envoie chaine de caractère
     IS EMIT ;                        \ restaure vecteur

 : EMET"   ( --- <chaine> en compilation ou interprétation)
     STATE @
     IF    COMPILE (EMET") ,"       \ compile (EMET")
     ELSE  ['] EMIT >BODY @ UP @ + @ >R
           \ prélève vecteur de EMIT
           ['] (RSEMIT) IS EMIT   \ remplace par (RSEMIT)
           34 PARSE >TYPE   \ envoie chaine de caractère
           R> IS EMIT           \ restaure vecteur de EMIT
     THEN  ; IMMEDIATE
 HEX
 : TOUCHE  ( n --- <mot> en compilation)
     CREATE ,
     DOES> @   13 >RS >RS ;
 41 TOUCHE ENVOI
 42 TOUCHE RETOUR
 43 TOUCHE REPETITION
 44 TOUCHE GUIDE
 45 TOUCHE ANNULATION
 46 TOUCHE SOMMAIRE
 47 TOUCHE CORRECTION
 48 TOUCHE SUITE
 49 TOUCHE CONN/FIN   49 TOUCHE CXF
 DECIMAL

 \ Procédures spéciales
 HEX
 : CONNEXION  ( ---)
     1B >RS  39 >RS  68 >RS  ;
 : DECONNEXION  ( ---)
     1B >RS  39 >RS  67 >RS  ;
 DECIMAL
 : ATTEND ( n ---) \ attente de n secondes
 0 DO 1000 MS LOOP ;
 45 FUDGE !
 DECIMAL
 EOF \ Fin du listing
```

MINITEL initialise le port de communication COM1 à 1200 bauds, 7 bits, parité E, DTR on, RTS off et valide l'interruption de signalement de réception de caratère.

>RS injecte un code ASCII vers le port de communication. Exemple, 65 >RS affiche 'A' sur le minitel; 12 >RS efface l'écran du MINITEL.

(RSEMIT) envoie un caractère avec transcodage des caractères accentués en fonction de l'état de la variable ACCENTS; ACCENTS ON transmet les caractères en format accentués; ACCENTS OFF transmet les caractères accentués en équivalents non accentués.

RSEMIT fait un peu double emploi avec (RSEMIT); envoie un caractère vers le port série et vers l'écran.

RSKEY? dépose un flag "vrai" si un caractère a été reçu par le port série.

RSKEY dépose le code ASCII du caractère reçu sur le port série. Identique à KEY, mais fonctionne seulement avec le port série.

(RSCHAR) insertion et traitement des caracteres reçus par le port de communication. Revectorise CHAR.

LOGIN donne la main au MINITEL en mode local. Pour interrompre la prise en main, taper LOGOUT depuis le MINITEL.

Ensuite, une série de fonctions permettent de se connecter à un service TELETEL (3615 SAM*JEDI pour exemple...) et de passer ses commandes et ses chaines de caractères depuis le clavier du PC. Je ne détaillerai pas ces mots à l'exception de:

EMET" envoi d'une chaine de caractères vers le minitel, mode local ou connecté: la chaine de caractère est compilation. Attention, la décompilation d'une définition contenant (EMET") perturbe l'affichage.

Les autres fonctions sont:

```
 ENVOI          RETOUR        REPETITION
 GUIDE          ANNULATION    SOMMAIRE
 CORRECTION     SUITE
 CONN/FIN ou CXF

 CONNEXION      DECONNEXION

 ATTEND  ( n ---) \ attente de n secondes
```

Afin de vous imprégner de l'intérêt des fonctions définies précédemment, voici comment gagner du temps (et de l'argent) à la connexion sur SAM*JEDI pour accéder au téléchargement de programmes:

```
 ( programme SAMJEDI d'accès à SAM*JEDI)
 CONNEXION
 5 ATTEND
 EMET" SAM*JEDI"
 ENVOI
 15 ATTEND
 EMET" [redacted]" ( ici votre pseudo)
 ENVOI
 5 ATTEND
 EMET" [redacted]" ( ici votre mot de passe)
 ENVOI
 11 ATTEND
 SUITE
 5 ATTEND
 EMET" 7"
 ENVOI
```

En pratique, compilez les définitions de gestion de la liaison série, puis tapez:

```
 : MNTL MINITEL HELLO ;
 ' MNTL IS BOOT
 SAVE-SYSTEM MINITEL
```

Ensuite, sous DOS, ou dans un fichier batch, tapez la ligne:

```
 MINITEL INCLUDE SAMJEDI
```

après avoir composé le 3615. Votre minitel se connectera automatiquement au serveur puis à l'option TELECHARGEMENT du menu général.

---

Le présent programme est disponible sur SAM*JEDI en téléchargement sous le nom RSINIT.FTH et sera intégré au module 4 de TURBO-Forth.

■

— FORTH —

## REDIRECTION DES ENTREES-SORTIES MS-DOS

par M.ZUPAN

Système: TURBO-Forth
Diffusion: 3615 SAM*JEDI et prochainement module M4 de TURBO-Forth.

LISTING: PIP.FTH

```
HEX
\ Primitives MS-DOS de redirection des E/S

CODE (FROM) ( hndl -- hndl fl )         \ duplique un ticket
     BX POP   45 # AH MOV   21 INT    AX PUSH
     U>= IF   0 # AX MOV    THEN 1PUSH   END-CODE

CODE (TO)   ( hndlfrom hndlto -- fl ) \ copie un ticket
     CX POP   BX POP   46 # AH MOV   21 INT
     U>= IF   0 # AX MOV   THEN 1PUSH   END-CODE
```

```
DECIMAL

0 CONSTANT KBD          \ ticket clavier
1 CONSTANT CON          \        écran
2 CONSTANT PROMPTER     \        sortie des  messages d'erreur
( écran )
3 CONSTANT AUX          \        auxiliaire
4 CONSTANT PRN          \        imprimante

VARIABLE <FROM \ conserve ticket from
VARIABLE FROM> \ conserve ticket copie from
VARIABLE TO>   \ conserve ticket to

: TO    ( <filename> hndl -- )
 \ redirige une sortie sur un fichier
  ?OPEN DUP <FROM !     \ sauve ticket d'origine
        DUP (FROM) ?DOS-ERR FROM> !     \ et sa copie
  ?OPEN PATHWAY         \ prend un nom de fichier
        DUP  0 (SEARCHO)  \ fichier existe déjà ?
        IF   1 (OPEN)     \ oui : on l'ouvre en écriture
        ELSE 0 (CREATE)   \ non : on le crée
        THEN ?DOS-ERR
  DUP 0 0 ROT 2 (SEEK) ?DOS-ERR  2DROP
   \ pointeur en fin de fichier
  DUP TO> ! SWAP (TO) ?DOS-ERR ;

: FROM  ( <filename> hndl -- )
 \ redirige une entrée depuis un fichier
  ?OPEN DUP <FROM !
        DUP (FROM) ?DOS-ERR FROM> !
  ?OPEN PATHWAY 2 (OPEN) ?DOS-ERR
  \ ouvre un fichier existant
  DUP TO> ! SWAP (TO) ?DOS-ERR ;

: RESTORE ( -- )    \ restitue direction d'origine
  FROM> @ DUP <FROM @ (TO) ?DOS-ERR
  (CLOSE) TO> @ (CLOSE) ;

' RESTORE CC @ CONTROL Z 2* + !
        \ RESTORE direct par Ctrl-Z

eof
```

MS-DOS 2.x et plus autorise les redirections d'entrées-sorties pour une grande variété d'applications utilisant les commandes > >> < ou | sous COMMAND.COM. Toutes ces opérations n'utilisent que deux fonctions primitives: la duplication d'un handle périphérique ou fichier et la copie d'un handle dans un autre.

Ces deux primitives sont données pour TURBO-FORTH dans (FROM) et (TO) à partir desquels il est aisé de définir tous les types de redirection pour des usages extrêmement variés. Les mots utilisateur TO FROM et RESTORE sont d'une syntaxe très simple. Voici quelques exemples:

CON TO ESSAI.TXT WORDS RESTORE
: crée un fichier ESSAI.TXT contenant la liste du vocabulaire courant. Il est préférable de faire au préalable ATTRIBUTS OFF pour éviter qu'il contienne les codes ANSI.

Une nouvelle séquence
CON TO ESSAI.TXT ..... RESTORE
: redirigera la sortie écran dans le fichier ESSAI.TXT à la fin de celui-ci.

PRINTING ON   PRN TO TEST.FTH
LIST TRUC  PRINTING OFF  RESTORE
: copie le fichier TRUC.FTH dans TEST.FTH avec écho du listing à l'écran. On utilise ici la double sortie écran-imprimante avec redirection de cette dernière dans un fichier.

CON TO PRN
: désactive l'écran: tout est envoyé sur l'imprimante jusqu'à l'exécution de RESTORE ou Ctrl-Z.

PRN TO AUX
: dirige toutes les sorties imprimante sur le périphérique auxiliaire, le minitel par exemple.

CON TO NUL INCLUDE TEST RESTORE
: interprète TEST.FTH sans aucune sortie sur l'écran. NUL est le périphérique nul qui comme CON PRN ou AUX est interprétable comme un fichier après TO.

KBD FROM TEST.FTH
: interprète TEST.FTH en direct, tel un fichier de macros. TEST doit se terminer par RESTORE ou ctrl-Z. On peut aussi définir un EOF temporaire équivalent à RESTORE si le fichier se termine par EOF. La redirection n'est pas protégée contre les erreurs: gare aux plantages avec un clavier totalement inactivé...

Très important: les redirections sont transmises par l'intégrateur !

KBD FROM MACROS.TXT PROGRAM A:\DOS\DEBUG.EXE RESTORE
: lance l'exécution de DEBUG avec les commandes inscrites dans le fichier MACROS.TXT. Prévoir ici une ligne 'Q' pour 'Quit' en fin de fichier...

PRN TO B:TEXTE.PRN PROGRAM WPROCESS.COM RESTORE
: lance un traitement de texte avec toutes les sorties imprimante dirigées dans un fichier TEXTE.PRN imprimable ultérieurement.

Ceci n'est qu'un aperçu des possibilités de redirections: fichiers de macros, transferts vers des drivers spécialisés, multi-redirections et filtrages, simulations etc...

## SOUNDS : sons sur PC
## programmation Forth des ports 42h 43h et 61h compatibles

par M. Zupan

Systèmes: TURBO-Forth et F83 Laxen et Perry MSDOS
Diffusion: Téléchargement 3615 SAM*JEDI et prochainement module M4 de TURBO-Forth.

LISTING:                SOUNDS.FTH

```
only forth  also definitions decimal

create NOTES
 \ table fréquence oscillateur 8284 / fréquence audio
( do    ré     mi     fa     sol    la     si     do  )
9121 , 8126 , 7239 , 6833 , 6087 , 5423 , 4831 ,
4560 , 4063 , 3619 , 3416 , 3043 , 2711 , 2415 ,
2280 , 2031 , 1809 , 1715 , 1521 , 1355 , 1207 , 1140 ,

  hex
: PLAY ( note durée -- )
  \ joue note 0 à 21 pendant une durée en ms
  swap B6 43 PC!       \ active le temporisateur 8253
  2* notes + dup
  C@ 42 PC! 1+ C@ 42 PC!
  \ programme le 8253 sur fréquence de la note
  61 PC@ 3 OR 61 PC!       \ active le haut-parleur
  MS                       \ attente en ms : régler FUDGE !
  61 PC@ FC AND 61 PC!     \ désactive le haut-parleur
; decimal

: TEST ( -- )
  dark
 ." Chacune des 22 notes doit être jouée ici pendant ½
seconde :"
    cr
 ." réglez la variable FUDGE (valeur actuelle = "
    fudge ? ." ) selon votre horloge interne" cr
  22 0 do I 500 play loop ;

test eof
```

Il n'y a pas plus piètre musicien que le PC: un malheureux haut-parleur dont on ne peut régler le volume sonore, un oscillateur rudimentaire sans enveloppe, des distorsions fréquentes, un port bloqué pendant l'exécution des sons (Ndlr: bref, une EXECUTION quasiment au sens propre...). C'est dire qu'il ne faut guère en attendre que de simples

messages sonores pour vos programmes...

Le LA de la 4ème octave du clavier est un son de 440 Hertz. En envoyant sur le temporisateur programmable 8253 l'ordre de diviser par 2711 la fréquence 1193180 Hertz de l'oscillateur 8284, on obtient une fréquence de 440 Hertz qui peut être envoyée sur le haut-parleur. Si cela vous chante (!), vous pouvez programmer les dièses et les bemols qui n'ont pas été rangés dans la table des diviseurs proposée ici. La durée de chaque note est progammée par la temporisation Forth en milli-secondes qui doit être ajustée selon la fréquence d'horloge interne du système au niveau de la variable FUDGE.

Paul Ortais
6 rue Pierre Curie
Verrieres le Buisson
91370 69.20.45.90

Appel au peuple du Jedi

Je suis ingénieur électronicien, designer de VLSIs, et en tant que tel sensible à la notion de performance. La simplicité a également le don de m'émouvoir. C'est pourquoi Forth me plait pour son potentiel, comme le NC4016 de Novix m'a sidéré à sa sortie il y a trois ans.

Il se trouve que:

- on développe le matériel (les chips) avec des outils fantastiques, et le logiciel est encore principalement produit par des méthodes primitives. Je reviendrai là-dessus pour ceux qui n'en seraient pas convaincus.
- Le NC4016 connait un succés bien inférieur à ce qu'il devrait être, bien que ce soit le micro le plus prometteur, et de loin, depuis longtemps.
- J'ai le projet personnel d'un chip du même genre, exécutant un Forth 32 bits, avec des facilités pour le traitement des bases de données, bien plus puissant que le 4016. Appelons le F32 pour faire court. Un centre de design de la région Parisienne est intéressé à titre amical pour fournir les outillages. Le financement n'est pas un problème bloquant, pour peu que le projet tienne la route.
- Il me faut l'aide de moustachus du Forth (ce que je ne suis pas) pour le coeur du problème, qui est: quel noyau va-t'on exécuter. En effet opter pour F83 serait parfaitement irréaliste. Il parait néammoins raisonnable de prévoir une compatibilité ascendante. Et puis, à quoi ressemblera un Forth 32 bits compatible, là, je suis un peu dépassé.

La meilleure façon de constituer une équipe de ce genre, c'est de constituer un comité à l'intérieur de l'association au lieu de vouloir que des acteurs se nomment. Soyons ouverts. Si quelque chose vous semble idiot dans mon papier, empressez-vous de me le dire!

J'ai naturellement une spécification de départ, mais il me parait trés prématuré de la détailler maintenant(Monsieur le Secrétaire, n'oubliez pas d'agrafer mes 2 feuilles dans le prochain numéro!).

Avant donc d'attaquer la falaise, je voudrais préciser mon objectif.

Essentiellement, utiliser le meilleur Forth sur les meilleurs circuits. maintenant, tout va trés vite. Les Risc arrivent, et même si leur emploi demande des compilateurs sophistiqués(ce qui n'est pas le cas pour le 4016) ils seront la normale dans deux ans. A ce moment, une application sérieuse nécessitera au minimum les 10 Mips dont on est loin aujourd'hui. Elle devra aussi:

- exploiter une mémoire >>100Mo virtuels, >16Mo physiques
- être au moins multi-tâches, souvent multi-utilisateurs
- être fondée sur une(des) base de données hiérarchisée
- exploiter des bibliothèques de procédures sur disques
- surtout, présenter à l'utilisateur un environnement de programmation orienté objet, c'est-à-dire avec encapsulation, classes et héritage, compilation retardée(à la limite, en temps réel vrai).

En y regardant bien, on voit que ces besoins forment un ensemble cohérent; que Forth s'adapte particulièrement bien à ces contraintes. Rien ne fera un meilleur objet qu'un mot Forth. Faire de la compilation retardée demande un compilateur simple... L'exploitation d'une base de données se ramène a gérer des listes et chaîner des pointeurs. C'est ce qu'une machine fera mieux que n'importe quoi d'autre.

F32/1 .../...

Dés lors faut-il prévoir à sa conception les facilités correspondantes. Le NC4016 ferait de l'excellent ouvrage, mais comme il n'a pas été optimisé dans ce but, on peut faire beaucoup mieux(un ordre de grandeur) encore.

Le mieux pour garantir l'encapsulation des procédures, c'est de ne pas permettre leur accés interne. Une interface de programmation graphique sera présentée un peu plus tard, çà me semble assez important pour être plus qu' un gadget: un impératif. Soit a créer une procédure donnant les racines d' un trinôme:

cliquer dans le browser sur Maths, puis Algèbre, puis Trinome. Racines ne s'y trouve pas(sait-on jamais, quelqu'un en a peut-être déjà eu besoin). Ouvrir l'éditeur d'icônes, dessiner

```
            ,______________
a  o--|a                  |
      |                   |
   o--|b  $%TOTO!   x1    |--o
      |                   |
      |             x2    |--o
   o--|c                  |
      |  v         s0 s1  |
       ________________
         |          |  |
         o          o  o
```

*Je lui donne un nom typiquement Forth*

nommer a,b,c les entrées, x1,x2 ls sorties. Pour les puristes, v valide la procédure, s0 et s1 sélectent le résultat. Ne pas oublier de nommer la procédure. Attention, si le mot n'est pas défini vous vous faites attraper et on vous ouvre l'éditeur de texte pour ecrire(enfin!) le Forth associé. Si ç'avait été déjà fait, le compilateur aurait simplement vérifié la cohé rence du dessin avec le mot.

Plus modestement, F32 devrait commencer par l'écriture d'une machine virtuelle, évidemment en Forth 83.

J'arrète là sinon j'en dirai trop. J'imagine qu'à présent vous vous di- tes "çà serait drôlement mieux avec les bonshommes de Léo Brodie, ces icônes là". Si c'est le cas, nous sommes d'accord pour laisser l'époque préhistorique du logiciel derrière nous.

Bien; maintenant que la bouteille est à la mer, laissons la voguer un peu. Nous continuerons par le tour des points critiques du projet: comment, pourquoi,quand,avec qui faire le F32;

et aussi conduite du projet
- le matériel qui est le gros oeuvre,
- le logiciel au coeur du problème,
- l'utilisateur qui est le but essentiel.
- vos critiques ardemment souhaitées...

F32/2

Le BROWSING ou "broutage", vertuchou!

Si vous utilisez une machine sérieuse pour faire un gros boulot, vous vous retrouvez comme un rien avec plusieurs milliers d'objets sur les bras.

Pour commencer, un bon millier de procédures de base pour l'OS, les entrées-sorties, un second OS, le graphique, l'arithmétique, le traitement des chaines... Ensuite il en faut encore autant pour le langage de haut niveau associé au noyau, pour celui que vous implantez en rab. Ne pas oublier l'application elle-même, qui sera la plus prolifique en la matière. Si plusieurs personnes partagent le système il faudra en outre installer au sommet du niveau "user" une hiérarchie qui, en fait, reproduit celle du service utilisateur(adieu, liberté chérie!).

Sous UNIX, on suppose à l'utilisateur une mémoire excellente, il n'a pas de mal à rejoindre son lieu de travail par:

```
CD /USER/SOL/PLANET/TER/HEMISPH/NORD/EUR/FRANC/VILLE/PARI/FOLIBERGER
```

çà se complique quand il veut recopier le fichier .../CIA dans ***/KGB:

```
COPY /USER/SOL/PLANET/TER/HEMISPH/NORD/AMER/USA/VILLE/LANGLEY/CIA TO
/USER/SOL/PLANET/TER/HEMISPH/NORD/ASIE/URSS/MOSCOU/KGB -Q
```

l'option -Q(query) est là pour lui demander "Ok pour recopier le fichier (nom-a-rallonge-1) dans (nom-à-rallonge-2)" ...on dépasse la ligne... puis rebelote pour demander si on peut effacer la version précédente, et enfin confirmer qu'on est parti de là, arrivé ici, ouf! On ne referait pas celà tous les jours. Et bien si, malheureusement.

La même manip avec un browser ressemble à ceci:

```
|TOP      |STAT     |WASHINGTON|PISCINE   |                  |Select     |
|USER     |CLIMA    |LOS-ANGELE|GOLF      |                  |Crée       |
|SOL      |RIVIER   |SAN-FRANCI|BILLARD   |                  |Supprime   |
|PLANET   |VILLE    |NEW-YORK  |RESTAURAN |                  |Ajoute     |
|TER      |ETHNO    |DALLAS    |EGLISE    |                  |Copie      |
|HEMISPH  |COMMERÇ  |MILWAUKEE |CENTRE-CO |                  |Déplace    |
|NORD     |INDIANS  |LANGLEY   |CIA       |                  |Insère     |
|AMER     |TREES    |TULSA     |COIFFEUR  |                  |etc.       |
|USA      |GEYSERS  |BATON-ROUG|MAC-DONAL |                  |           |
|         |         |DENVER    |AIRPORT   |                  |           |
|----------------------------------------------------------|-----------|
|                                                          |           |
|                                                          |Clean_Lib  |
|                                                          |Mizajour   |
|                                                          |Protection |
|----------------------------------------------------------|-----------|
|SOL      |SIBERIE    |KREMLIN|                            |Select     |
|PLANET   |MOSCOU     |BOLCHOI|                            |Crée       |
|TER      |           |KGB    |                            |Supprime   |
|HEMISPH  |           |       |                            |Ajoute     |
|NORD     |           |       |                            |Copie      |
|ASIE     |           |       |                            |Déplace    |
|URSS     |           |       |                            |Insère     |
```

Cliquer sur un truc pour l'attraper, le caser où vous voulez, l'insérer, si c'est un peu plus compliqué tout mettre temporairement dans la zone médiane. Si on veut manipuler des paquets, des catégories: cliquer dessus pour les allumer, ensuite transbahuter. Quand tout a l'air nickel, renvoyer la fenêtre au néant, le ménage est fait. Temps nécessaire pour réorganiser trois mille fichiers dans 100 catégories sous une douzaine de niveaux: une bonne journée de travail. Sous Unix ou DOS ou xxx: infaisable.

F32/3

COURRIER: INITIATION

Tout d'abord, je vous écris pour vous commander TURBO-Forth avec M1+M2...

Bravo pour la traduction de TURBO-Forth dans les langues européennes. Avez-vous pensé à créer un tutorial sur disquette et livre (ou brochure). Il permettrait au béotien de se familiariser et apprécier sa puissance et son esprit. D'abord victime d'un coup de foudre pour F83, cette envie de programmer dans un langage ésotérique pour franc-maçons de l'informatique s'est estompée rapidement. Malgré l'achat de votre livre "FORTH-83 Standard"qui mélange allégrement les instructions (sans pratiquement d'exemples) et les vecteurs, je me suis rendu compte que je n'avais pu assimiler que les instructions simples. Les adresses de variables, les manipulations de vocabulaire (ex: ALSO, ONLY...), les fichiers, la méta-compilatipon font la singularité et l'efficacité du FORTH, mais hélas font aussi le désespoir du débutant forthien.

Une rubrique régulière pour nous les débutants forthiens avec une démarche complète de pratique (chargement, compilation, commandes intermédiaires) expliquées en détail permettrait d'accrocher, de sauver et de donner un aperçu au nouveau forthien.

La revue, actuellement est faite pour les fortiens chevronnés et si cela continue ainsi, je retournerai définitivement dans l'univers de dBASE et de mon cher BASIC compilé.

Je contribue néanmoins avec mon obole en espérant que mon sauvetage et que la promotion du FORTH soit réussie.

Je trouve les articles moins morcelés, les sujets plus universels et l'encadrement des titres une nécessité heureusement adoptée.

Si une publicité sur le TURBO-Forth est disponible sur disquette, imprimable sur imprimante graphique IBM, pouvez-vous me la procurer? J'aimerai tant diffuser TURBO-Forth dans les clubs informatiques de la région caennaise.

Tant de mois écoulés à m'enliser dans le F83 m'ont rendu trop bavard.

J.F. MARIE 14490 BALLEROY

REPONSE:

Bavard, vous? Oh non! S'il y avait plus de lettres comme la votre, nous aurions depuis longtemps créé une rubrique pour débutants. Mais comme nous ne pouvons appréhender le niveau de connaissance de nos lecteurs, nous ne savons par quel bout commencer. Pour mieux situer les différences de niveau, disons qu'il y a des programmeurs (comme vous) qui débutent et ont besion du B.A.BA, et d'autres qui une fois reçu TURBO-Forth, se mettent à pondre des routines et des programmes de haut niveau (comme le simulateur de CPU, JEDI 44 p 15). Mais que vous soyez débutant en FORTH ou rtompu à toutes les astuces de programmation, vous avez tous des questions concernant TURBO-Forth portant sur des points précis.

Donc, nous pouvons commencer une rubrique "débutant FORTH", et la compléter par une rubrique "clinique" où on tenterait de résoudre les difficultés individuelles.

Pour rendre plus efficace l'information et répondre rapidement aux questions, n'hésitez pas à utiliser le service 3615 SAM*JEDI et à déposer votre prose sur le FORUM. Selon la difficulté de la qusetion, vous aurez une réponse dans un délai de un jour à une semaine environ (excepté quand on part en vacance...).

Nous n'avons pour le moment aucune publicité sur TURBO-Forth, mais elle ne devrait tarder à être mise en page.

---

COURRIER: FORTH MACH II

Pouvez-vous m'indiquer où il est possible de trouver en France le Forth MACH II pour ATARI ST.

O. DUPONT 59300 VALENCIENNES

REPONSE:

Si nous le savions nous-même, nous ne l'aurions pas caché. Malheureusement, nous ne pouvons que vous donner les références U.S.:

| | | |
|---|---|---|
| MACH II MacINTOSH | 99.95$+10.00$ | port |
| MACH II ATARI ST | 99.95$+10.00$ | port |
| MACH II pour OS-9 | 495.00$+10.00$ | port |

Mac INTOSH: inclus virgule flottante, multi-segment, gestion souris, RMaker, manuel de 500 pages.

ATARI ST: GEM et TOS, virgule flottante, redirection I/O, gestion souris, manuel de 300 pages.

OS-9: génère un code relogeable, support toutes entrées OS-9, extensions OS-9 permanentes. Liens entre FORTH et C et inversement. Inclus un manuel de 400 pages.

Commande: PALO ALTO SHIPPING COMPANY P.O. Box 7430 Menlo Park, California 94026. (USA)
Tel support: (19-1) 415 / 854-7994
vente: (19-1) 800 / 44FORTH

Pour payer: VISA/MC accepté. Par chèque, consulter votre banque qui émettra un chèque de banque en $US à joindre à votre commande. Dans ce cas, écrivez en recommandé, le courrier s'égarant parfois... Les mandats internationaux ne sont pas acceptés aux USA.

---

COURRIER: NOVIX

Des articles sur FORTH, bien entendu, mais aussi sur des nouveaux systèmes performants, tels que le NOVIX (dot vous avez parlé dans le n° 41, et le nouveau MICROWAY Multiputer à 1500 Mips (décision informatique de cette semaine). Bravo pour la nouvelle police de caractères (le C ne se confond plus avec le crochet [).

L. NGUYEN-HUU 75014 PARIS

REPONSE:

Si vous signalez un article, joignez une photocopie. La rédaction de JEDI n'a pas toujours le temps de relever les informations du type de celle dont vous nous faites part. La nouvelle police de caractère est l'oeuvre de l'imprimante à laser BROTHER.

TELEMATIQUE

## ACCES AU SERVICE TELEMATIQUE DE JEDI

par Marc PETREMANN

Comment joindre le secrétaire quand son téléphone est occupé ou qu'il ne répond pas? Peut-on déranger un rédacteur d'article à 2 heures du matin pour avoir des éclaircissements sur une routine délicate? Que faire quand on sèche sur un mot FORTH?

Jusquà présent, il n'y avait qu'une solution: écrire et attendre que votre prose passe dans la rubrique courrier, notamment si votre question est d'intérêt général. Le délai: deux à trois mois. Ce n'était pas la solution la plus efficace pour qui est habitué à gratter les micro-secondes dans des boucles optimisées.

Avant JEDI, il n'y avait rien! Puis vint JEDI pour étancher votre soif de FORTH! Maintenant il y a:

SAM*JEDI

Huit caractères magiques qui donnent un second souffle à notre association.

VOTRE MATERIEL:

Pour nous contacter, un MINITEL suffit. Mais si vous disposez en plus d'un PC et d'un câble de liaison MINITEL-PC, c'est mieux. On n'interdit pas non plus les liaisons entre MINITEL et APPLE/AMSTRAD/ZX81/VAX/SUN...CIA/KGB... pourvu que votre liaison soit gérée par un bon logiciel de gestion de communication.

Des logiciels de gestion il y en a des tas. Rien que sur PC, ils fleurissent les bacs à disquette du fouineur invétéré s'approvisionnant par petites annonces gratuites dans des revues de seconde zone (autres que JEDI...).

Ce que doit faire le logiciel de gestion: capturer et enregistrer la communication avec le serveur. On avait déjà cité dans un précédent numéro le logiciel LCE-COM. Mais pour saisir les fichiers du téléchargement, un utilitaire du type PROCOMM est plus adapté:

- capture de pages au format VIDEOTEX

- capture ASCII pour le téléchargement

Dans le second cas, la capture ne peut se faire page par page, mais en continu, les logiciels à télécharger étant envoyés sans discontinuité ni attente de contrôle en mode ASCII 7 bits.

Pourquoi capturer les communications:

- pour gagner du temps. Un programme diffusé dans JEDI n'est disponible que 3 à 6 mois plus tard dans un module additionnel de TURBO-Forth; le recopier à la main est rebutant et source d'erreur. En téléchargement, il est disponible de suite. La capacité d'accès du serveur assure une disponibilité permanente.

- pour économiser de l'argent. Consulter un FORUM et prendre des notes fait perdre du temps et incrémente votre compteur de taxes téléphoniques. En capturant le contenu du FORUM, vous pourrez le relire tranquillement une fois déconneté.

NOTRE MATERIEL:

Le serveur SAM*JEDI est installé sur un système de type PC-AT installé à PARIS, au siège social de la Société VICTEL, 14 bd Montmartre.

Nous bénéficions de la commission paritaire de SEQUOIA PRESSE, 4ter rue de l'Abbaye, MEUDON, permettant ainsi notre diffusion sur le 3615.

Le système PC-AT tourne sous DOS QNX, système de gestion multi-tâches. Il est relié à un modem à haut débit via la liaison série, puis par liaison permanente 4 fils entre le modem et le central BONNE NOUVELLE.

Le système de gestion QNX associé au logiciel serveur assure une capacité d'accueil de 32 voies simultanées, qui sera portée à 64 voies simultanées prochainement.

La capacité de stockage est de 72 Mo répartie sur deux disques durs.

Outre SAM*JEDI, le serveur assure le trafic des services SAM et SAM*ATA entre autres.

NOTRE VOCATION:

Le serveur SAM*JEDI est essentiellement destiné à assurer un service d'assistance logicielle aux utilisateurs de FORTH (F83 CP/M ou MSDOS et TURBO-Forth).

Ensuite, le serveur permet le dialogue entre utilisateurs désirant travailler sur un projet commun. Dans ce cas, SAM*JEDI remplit la fonction de répondeur ou de télex.

Enfin, par l'attribution de boîtes aux lettres, il permet un dialogue confidentiel pour toute communication entre programmeurs ne souhaitant pas en faire part dans le FORUM.

Donc, une vocation essentiellement professionnelle, mais accessible à tous, adhérents ou non, débutants ou programmeurs chevronnés. Avec le temps, le style de SAM*JEDI évoluera certainement, mais restera une mine de renseignement pour tous ceux qui pratiquent et développent à l'aide de FORTH.

LA MAINTENANCE:

Le serveur SAM*JEDI dispose d'une voie d'accès par 3614, avec un code réservé à la maintenance. Celle-ci est assurée par deux personnes:

- Marc PETREMANN, secrétaire de l'association, pour répondre sur le FORUM.
- Erik FORGET, pour la mise à jour des BAL et des messages du FORUM.

La maintenance cafouille encore un peu, mais à terme, nous serons en mesure de faire le nettoyage sur le FORUM et dans les BAL. Une BAL inutilisée trop longtemps et vide sera supprimée. Les messages sans intérêt dans le FORUM seront éliminés.

Les programmes à télécharger sont fournis directement à VICTEL sur disquette au format ASCII MSDOS. Ils sont disponible dès la copie sur le serveur:

- 12h00, chargement sur serveur
- 12h02, mise à jour du menu de sélection
- 12h05, validation du nouveau menu; les logiciels peuvent être téléchargés par n'importe qui et par plusieurs consultants simultanément.

Les modifications du logiciel serveur sont à la charge de la société VICTEL qui ne demande aucune conterpartie financière pour notre hébergement en dehors des revenus provenant des connexions au serveur. Le coût jour/homme pour la mise en place de SAM*JEDI est estimé à plus de 20000 Fr actuellement. Il faudra plus de 600 heures de connexion pour amortir cet investissement intégralement pris en charge par VICTEL.

COMMENT ACCEDER A SAM*JEDI:

D'abord disposer d'un MINITEL (ou d'une carte de communication adéquate, type KORTEX ou WINNER'S). Ensuite, avoir le téléphone. Enfin, composer le 3615. Et pour finir, taper SAM*JEDI.

Ah, vous faites partie des adhérents n'habitant pas en France (si, si, il y en a..): composer le (33) 36.43.15.15 puis SAM*JEDI.

Attendre un peu, le logo JEDI s'affiche, très beau, très sobre et en couleurs, puis vous demande votre identification:

- c'est votre première connexion: donnez-vous un pseudo, ou mieux, votre nom. Evitez JHCHJF, ou VENDREDI13, vous ne serez pas pris au sérieux. Si votre nom est assez répandu, rajoutez l'initiale ou le prénom complet: PASCAL A sera accepté. Ensuite, entrez le code d'accès: ce code est strictement confidentiel et garantit l'inviolabilité du contenu de votre BAL.

- ce n'est plus votre première connexion. Vous avez votre BAL; tapez votre pseudo et votre code d'accès.

Taper sur SUITE pour accéder au MENU JEDI:

B i e n v e n u e sur S A M * J E D I

1 Vous ouvrir une BAL-JEDI

2 Ecrire un message

3 Lire votre bal

4 Lire le forum

5 Consulter l'annuaire

6 Relire d'anciens messages

7 TELECHARGEMENT

Tapez votre choix N .. + ENVOI

Utilisation de SAM*JEDI GUIDE

L'option GUIDE n'est pas à jour actuellement. Les autres sélections sont:

1- pour ouvrir une BAL-JEDI; cette option permet la création d'une BAL personalisée JEDI et indépendante des BALs ou pseudos gérés sur les autres services de SAM.

2- pour écrire un message; cette option sera expliquée en détail plus loin.

3- lire votre BAL; on vous a écrit personnellement. Vous pouvez consulter votre BAL. En fin de consultation, un

message vous informe:

EFFACEMENT DU COURRIER O/N
Si vous répondez par O (majuscule), votre boite aux lettres sera vidée. Cette option permet de conserver ou non le courrier consulté pour une connexion ultérieure.

4- lire le FORUM: c'est ici que vous pourrez consulter tous les messages publics contenant des questions, réponses à questions, trucs, astuces de programmation.

5- consultation de l'annuaire: avant d'ecrire un message confidentiel, regardez si votre correspondant a toujours sa BAL. S'il a eu l'intelligence de mettre son om en clair, vous le retrouverez facilement. Exemple, Monsieur M a écrit un article génial, mais la routine RQUEQUECHOSE pose problème sur votre TRUC-PC; son article est signé; cherchez son nom sur SAM*JEDI, option 5; oui, il existe; vous pouvez lui écrire personnellement.

Voici le contenu de l'annuaire au 18/07/88:

ANNUAIRE JEDI

1 ALPO
2 COURTOIS
3 DIMANCHE17
4 ESTHETE
5 EUREKA
6 FORTH7
7 JACCOMARD
8 JMCFORTH
9 JMCO
10 JNV
11 JUNG
12 LAMBERTPH
13 LAVEAU
14 MOHR
15 PAUL ORTAIS
16 PILVERDIER
17 SECRETAIRE

Quitter SOMMAIRE   Autres pseudos SUITE

Le pseudo du secrétaire de l'ASSOCIATION est SECRETAIRE évidemment.

6- relecture d'anciens messages; ceux du FORUM naturellement. A chaque consultation, votre date et heure de connexion sont enregistrées. La lecture du FORUM affichera tous les nouveaux messages reçus en votre absence. Mais entre-temps vous vous êtes équipé du merveilleux logiciel de capture XYZABCPM et vous voulez avoir une trace de ce qui s'est dit depuis une date donnée ou depuis l'ouverture du FORUM; tapez la date choisie et lancez la relecture. La date est au format JJMMAA et non JJ/MM/AA. L'appui sur ENVOI sans préciser de date de départ relance la lecture de tout le contenu du FORUM.

7- TELECHARGEMENT: l'option maitresse de SAM*JEDI. Tout programme un peu important pour TURBO-Forth est disponible dans cette option. Choix du langage:

| | |
|---|---|
| | L'association |
| LE | J E D I |
| TELECHARGEMENT | vous propose |
| | de nombreux |
| | logiciels. |

| | |
|---|---|
| 1 FORTH | Infos pour le |
| 2 PROLOG | téléchargement |
| * BASES DE DONNEES | tapez GUIDE |
| * PASCAL | |
| 5 JEUX | Votre choix |
| 6 DIVERS | N 1 + ENVOI |

Les prog de type J sont de l'ass. JEDI

Seul FORTH et PROLOG contiennent pour le moment des programmes. Voici le menu des programmes FORTH:

F O R T H   P C

| N | Programme | Type | | Description |
|---|---|---|---|---|
| 1 | SIMU1802.FTH | J | A | SIMUL MP1802 RCA |
| 2 | UT4.S02 | J | B | SIMUL MP1802 RCA |
| 3 | WINDOWS.FTH | J | A | MULTI FENETRAGE |
| 4 | DBCOOK.FTH | J | A | ACCES FICH DBASE3 |
| 5 | ALEPH.FTH | J | A | VARIABLES LOCALES |
| 6 | COMPACT.FTH | J | A | COMPACTEUR T-F83 |
| 7 | ERATHOS.FTH | J | A | CRIBLE ERATHOSTHEN |
| 8 | FLOAT.COM | J | B | MODULE PRECOMPILE |
| 9 | FLOAT.FTH | J | A | SOURCE FTH+PASCAL |
| 10 | FPACK.FTH | J | A | VIRG FLOT T-F83 |
| 11 | HALT.FTH | J | A | REND TF83 RESIDENT |
| 12 | HERCULE.FTH | J | A | FONCT. GRAP.HERC |
| 13 | HERCULA.FTH | J | A | EXMP GRAPHIQUES |
| 14 | INFIX.FTH | J | A | NOTATION INFIXEE |
| 15 | EDIT.COM | J | B | EDITEUR ASCII TF83 |

N du logiciel à Télécharger :..ENVOI
SOMMAIRE ou autres programmes   SUITE

Pour la première page. Suite du menu pour la seconde page:

F O R T H   P C

| N | Programme | Type | | Description |
|---|---|---|---|---|
| 1 | EDITERR.MSG | J | A | MESSAG-ERR EDITEUR |
| 2 | FORTH.VOC | J | A | AUTODOC FORTH TF83 |
| 3 | ROOT.VOC | J | A | AUTODOC ROOT TF83 |
| 4 | TURBO.COM | J | B | TURBO F83-STANDARD |
| 5 | SWORDS.FTH | J | A | TRI VOCABULAIRE |
| 6 | TIME.FTH | J | A | FONCTIONS CHRONOM. |
| 7 | TIMER.FTH | J | A | HORLOGE SYSTEME |
| 8 | UFILES.FTH | J | A | OUTILS FICH. TF83 |

Pour chaque page, le logiciel est marqué par un numéro, le nom du fichier, la lettre J, la lettre A ou B et le descriptif du programme. Pour télécharger un programme, procéder comme suit:

- mettre en route le programme de capture TELETEL
- accéder à SAM*JEDI, option 7
- choisir son programme dans une liste; si le programme est marqué A, une capture ASCII sera suffisante; dans le cas contraire, utiliser TELECHAR sur disquette KERMIT PC. La lettre J indique que le programme est d'origine JEDI, la société VICTEL se réservant le droit de rajouter des programmes à sa convenance sans lettre J.

ECRITURE ET ENVOI D'UN MESSAGE:

Choisir l'option 2, que ce soit pour envoyer un message public ou confidentiel.

La saisie du message peut commencer immédiatement après la sélection de cette option.

Le message est composé de 20 lignes de moins de 40 caractères. Les caractères de contrôle VIDEOTEX ou accentués ne sont pas acceptés pour le moment. Ceci provient du programme de gestion TELETEL, mais sera arrangé dès que possible. En cas d'erreur, appuyer sur CORRECTION.

Pour passer à la ligne suivante, taper SUITE; à la ligne précédente, taper RETOUR.

Arrivé à la 20ème ligne, si vous voulez continuer votre message, taper ENVOI et saisir une nouvelle page.

Pour achever la saisie d'un message, mettre un point en début de ligne suivante et taper sur ENVOI.

Ensuite, un menu vous propose:

ENVOI DU MESSAGE:

1. SYSOP JEDI

2. FORUM

3. CONFIDENTIEL

L'option 1 est à réserver pour communiquer avec la gérance de SAM.

L'option 2 envoie votre message sur le FORUM.

L'option 3 affiche l'annuaire; taper le numéro précédant le pseudo de votre correspondant. Le même message peut être envoyé à plusieurs correspondants; il suffit de taper les numéros précédant chaque pseudo suivi de ENVOI.

A chaque phase du déroulement du service, l'appui sur SOMMAIRE renvoie au choix ou au menu précédent.

L'abandon de SAM*JEDI se fait par appui sur CONNEXION/FIN.

VOS SUGGESTIONS:

Vous avez une idée de modification ou d'amélioration du service SAM*JEDI, faites-nous en part. Nous répercuterons votre proposition sur VICTEL.

TELEMATIQUE

## CONTENU DU FORUM SAM*JEDI AU 11/07/88

par M. PETREMANN

Pour que les adhérents de JEDI ne disposant pas d'un MINITEL ne se sentent pas rejetés, voici une transcription du contenu du FORUM reprenant les propos intéressants diffusés depuis la mise en service.

--------------------

SECRETAIRE 21.06.88 10h37
Enfin, la messagerie JEDI ouvre ses portes. Vous pouvez dorenavant poser des questions d'ordre technique sur tous les problemes d'ordre technique que vous rencontrez avec TURBO-Forth et tous les autres langages que vous pratiquez. Vous etes cordialement invites a aider les autres dans la mesure de vos competences.

--------------------

SECRETAIRE 28.06.88 16h07
TURBO-Forth est maintenant disponible en version anglaise au meme prix que la version francaise:
37,00 Fr le module M1ou M2 ou M3
70,00 Fr pour deux modules
100,00 Fr pour les trois modules port compris.

--------------------

De SECRETAIRE Du 05.07.88 A 12h51
SAISIE DES MESSAGES SUR SAM*JEDI
On me signale deja quelques difficultes de saisie de messages:
- les accentues perturbent la saisie; ne pas les utiliser. Si vous en tapez une accidentellement, tapez sur ANNULATION et resaisissez la ligne.
- une ligne ne peut depasser 39 caracteres.
- un message est decoupe en pages de 20 lignes; la taille d'un message n'est pas limitee.
- passer de ligne en ligne en tapant sur SUITE.
- pour finir un message, mettre un point en debut de ligne suivante et taper sur ENVOI.

POUR CHOISIR SON CORRESPONDANT,
attendez de revenir au menu proposant:
1 SYSOP JEDI
2 FORUM
3 CONFIDENTIEL
L'option 1 est a reserver pour communiquer avec la gerance de SAM.
L'option 2 diffuse votre message dans le FORUM; il peut etre lu par tout le monde.
L'option 3 vous affiche une liste de correspondants; choisissez le correspondant a qui vous destinez votre message; profitez-en, vous pouvez envoyer votre message a plusieurs personnes en tapant successivement les numeros de vos differents correspondants.

N'ABUSEZ PAS DES MESSAGES SANS INTERETS
Evitez de deposer dans le Forum un message sans signification; il sera supprime sans pitie. Idem si votre message ne concerne pas la micro: allez vendre votre voiture ailleurs.
Ne vendez pas non plus de logiciel(s) pirate(s) et ne cherchez pas l'ame soeur pour une nuit de folie: il y a d'autres serveurs pour cela.
Le FORUM JEDI est un outil; vos suggestions sont les bienvenues; les petits defauts de demarrage seront arranges en temps utile.
Amities.

--------------------

De SECRETAIRE Du 05.07.88 A 12h58
CORRECTION DE UM/MOD
Une bogue dans UM/MOD subsiste dans la version F83 de Laxen et Perry et a ete repercutee sur TURBO-Forth. Mr Fred BEHRINGER, notre correspondant Outre-Rhin nous communique la solution:

```
 ' UM/MOD 30 DUMP
affiche
 *******\/  9  A  B  C  D  E  F
         7A 07 5B 5A 58 39 DA 7C
                               :
        que l'on remplace par 72
```

Et la version en code machine peut etre modifiee dans le programme source (fichier KERNEL.TXT) par la nouvelle version de UM/MOD decrite en page suivante:

```
CODE UM/MOD  ( code 8086)
  bx pop  dx pop  ax pop
  bx dx cmp
  U)=  ( au lieu de  >=   )
  IF -1 # ax mov   ax dx mov
     2push
  THEN
  bx div
  2push END-CODE
```

--------------------

De SECRETAIRE Du 05.07.88 A 13h03
MICRO-PROCESSEUR FORTH FRANCAIS
Mr Paul ORTAIS envoie un appel au peuple du JEDI:
- Il a le projet personnel d'un chip similaire au NC 4016 de Novix, executant un Forth 32 bits avec des facilites pour le traitement de bases de donnees.
- Un centre de design de la region parisienne est interresse a titre amical pour fournir les outillages. Le financement n'est pas un probleme bloquant, pour peu que le projet tienne la route.
- Il faut l'aide de moustachus du Forth pour le coeur du probleme, qui est: quel noyau va-t-on executer. A quoi ressemblera un Forth 32 bits?
Pour en savoir plus longuement, attendez la publication dans JEDI 45 des trois feuillets qu'il m'a fait parvenir, ou contactez le a:
Mr PAUL ORTAIS
6, rue Pierre Curie
91370 VERRIERES LE BUISSON
tel dom: 69.20.45.90

--------------------

De JACCOMARD Du 05.07.88 A 16h42
A SECRETAIRE
OUI' IL EST BIEN LE SERVEUR. OUI! J'AI RECU LE NO 44 DE JEDI MAIS JE NE SUIS PAS GERMANOPHONE
Y A-T-IL UN TRADUCTEUR DANS LE FORUM???

--------------------

De SECRETAIRE Du 08.07.88 A 13h21
MICRO-PRO 32 BITS FORTH
L'idee de Mr Paul ORTAIS est excellente. Outre le fait qu'elle mobiliserait les competences autour d'un projet ambitieux, elle permettrait d'eprouver les connaissances et les limites du FORTH.
Pour ma part, je propose la demarche suivante:

1) definition du jeu d'instructions du micro-processeur

suite page 26

# FORTH
# TO THE FUTURE

*MITCH BRADLEY-MOUNTAIN VIEW, CALIFORNIA*

32-bit machines are here to stay. Over the next few years, 32-bit machines will grow in importance. Forth must be able to use the full power of 32-bit machines.

This article presents a consistent, proven scheme for using Forth on 32-bit machines, based on several years of experience with 32-bit systems. It does not address the problems of simulating extended addressing on the 8086. The focus is on making the transition from 16-bit Forth systems to "real" 32-bit architectures like the 80386, the 68000, and the IBM RT.

**Goals**

1. Programs should run unchanged on either 16-bit or 32-bit machines.

2. The 32-bit machine must not be penalized. The full power of the 32-bit machine must be available.

**Tradeoffs**

1.Existing programs may have to be modified in order to make them run on either size machine.

2. A lot of new words are specified. These words are necessary because the existing words do not work right on 32-bit machines.

**Justification**

Forth will not succeed if it remains stuck at 16 bits while the world switches to 32 bits. Insisting that existing programs run unchanged on 32-bit machines penalizes the 32-bit implementation.

The wordset presented here penalizes neither 16-bit nor 32-bit machines. It adds no new funtionality, it simply specifies a set of names for words whose behavior is independent of the machine size.

**What is a 32-bit Machine?**

The distinguishing factor is the size of the address arithmetic. The address arithmetic determines the size of an address that can be easily calculated. The 68000 is a 32-bit machine — even though its data path is only 16 bits wide, and even though the package has only 24 address pins — because it is easy to calculate 32-bit addresses. The 80286 is a 16-bit machine, even though it has more than 16 address pins; addresses outside a 16-bit bank are painful to calculate.

Forth prefers to represent an address as a single entry on the stack, since the same operators are used for both number arithmetic and address arithmetic. It is possible, but troublesome, to represent addresses as multiple stack items. The preferred width of the Forth stack on a particular machine is the size of that machine's address arithmetic.

**Compatibility Problems**

When moving code from a 16-bit Forth implementation, there are two major problems.

Most Forth programs contain lots of things like `2+` and `6+`. This is fine if you are trying to add two or six to a number, but it causes problems if you are trying to increment an address to point to the next number. On most 32-bit machines, successive numbers are four addresses apart, not two.

16-bit numbers are inadequate for many purposes, so Forth has "double numbers," which are 32 bits, represented as two stack items. 32-bit systems do not need two stack items to represent a 32-bit number. Existing 16-bit programs use fancy stack manipulations to move the separate halves of a 32-bit number. Double-number operators like `2DUP` are used both for pairs of single numbers and for 32-bit numbers. A 32-bit number is an entirely different thing than a pair of numbers. They just happen to have a similar representation on a 16-bit system. On a 32-bit system, this isn't true.

**Solutions**

Some brief words about the nomenclature used here:

A "normal" is a number that is represented as one stack entry. On a 16-bit machine, a normal is 16 bits. On a 32-bit machine, a normal is 32 bits. The majority of all Forth operations are performed on normal numbers.

A "longword" is always a 32-bit number. On a 16-bit machine, a longword is represented as two stack entries. On a 32-bit machine, a longword is represented as one stack entry.

A "word" is always a 16-bit number. On a 16-bit machine, a word is represented as a single stack entry. On a 32-bit machine, a word is represented as the low 16 bits of a stack entry, with the upper 16 bits set to zero.

A "character" is always an 8-bit number. A character is represented as the low eight bits of a single stack entry, with the remaining upper bits set to zero.

**Address Incrementing Words**

Changing all occurrences of `2+` to `4+` doesn't solve the problem, it just sweeps it under a different rug. What we really need is a way to increment an address by the right number, regardless of what machine we're on. To do this, we define some names for the sizes of things.

`/N` ( -- n ) "per-n"
The number of bytes in a "normal" number, which is a single stack entry. `/N` is four on a 32-bit machine and two on a 16-bit machine.

`/L` ( -- n ) "per-l"
The number of bytes in a 32-bit "longword." `/L` is four on all machines.

`/W` ( -- n ) "per-w"
The number of bytes in a 16-bit "word." `/W` is two on all machines.

`/C` ( -- n ) "per-c"
The number of bytes in an 8-bit "character." `/C` is two on all machines.

The notation `/X` for "the number of bytes in an X," pronounced "per-x," follows the recommendations in Kim Harris's nomenclature guidelines ("Forth Coding Conventions," *Proceedings of the 1985 FORML Conference*).

You might think, since `/L` is always four, that the name "`/L`" is not needed. Ditto for `/W` and `/C`. However, the symbolic name `/L` clearly indicates that the code is dealing with the size of a longword, rather than the number four, which could be anything — perhaps the expected number of legs on a cow. Magic numbers make programs harder to understand and maintain!

Others have suggested the names `CELL` and `LSIZE` instead of `/N`; however, the name "normal" and the mnemonic "N" will be useful to us later.

What will we do with these constants? One obvious answer is to replace occurrences of `2+` with `/N`. Similarly, in cases where we want to step through an array of 16-bit words or 32-bit longwords, we might use `/W+` or `/L+`.

Another use is to calculate the number of bytes to **ALLOT** for some data structure or array. For instance, if we need space for 100 normal numbers, we could write **100 /N * ALLOT** instead of **100 2* ALLOT**.

A third use is to index into an array. Instead of writing the code in Figure One-a, for instance, we might use instead the definitions in Figure One-b. Notice that in all these cases, we have given up some efficiency! The word **2+** probably executes faster than the two words **/N +** and **2*** almost certainly is faster than **/N ***. We will not tolerate such inefficiency! Therefore, we define some more words, their existence amply justified by frequent use.

The functions in Figure Two are presently performed with **1+**, **2+**, and **4+**, whose use does not work on all machines. Most occurrences of **2+** in existing Forth code can be replaced by **NA1+** to make the code more transportable. The names stand for "normal-address-one-plus," etc., indicating that they increment an address to the next datum of a particular type.

Some machines do not directly address bytes. For instance, the Novix Forth chip is a word-addressed machine. Adding one to an address moves to the next 16-bit word, not to the next byte. For such machines, **NA1+** is not equivalent to **/N +**. The real rule is that **NA1+** should increment an address to point to the next item of a given type.

The words in Figures Three and Four find the address of the nth item in an array of items starting at addr. For instance, **NA+** is equivalent to **/N* +** on most machines.

This may seem like a lot of words. It is a lot, but they are frequently used, which is the same justification used for words like **1+** and **2***.

### Explicit 32-bit Operators

One solution to the double-number problem on 32-bit machines is to make double numbers 64 bits. This is attractive because it is compatible with existing code that manipulates double numbers as pairs of stack entries. On the other hand, it is inefficient. 64-bit arithmetic is slower than 32-bit arithmetic on most machines. While many applications require more than 16 bits of precision, few require more than 32.

I believe the best long-term solution is to define a set of words that explicitly operates on 32-bit data, regardless of the machine's word size. The names of these words begin with the letter L, indicating that they operate on "long" operands. Their implementation is simple. On a 16-bit machine, they are the same as the existing "D" words (e.g., **D+**) and the "2" words (e.g., **2DROP**). On a 32-bit machine, they are the same as the regular single-number operators. The important point is that the "L" operators *always* operate on 32-bit longwords, regardless of machine size.

### Long Arithmetic Operators

Some 32-bit arithmetic operators:

**L+** ( L1 L2 -- L3 ) "l-plus"
Adds 32-bit longwords. On a 16-bit machine, **L+** is the same as **D+**. On a 32-bit machine, **L+** is the same as **+**.

**L-** ( L1 L2 -- L3 ) "l-minus"
Subtracts 32-bit longwords. On a 16-bit machine, **L-** is the same as **D-**. On a 32-bit machine, **L-** is the same as **-**.

**L*** ( L1 L2 -- L3 ) "l-times"
Multiplies 32-bit longwords. On a 16-bit machine, **L*** is the same as **D*** (which is not included in the standard). On a 32-bit machine, **L*** is the same as *****.

**L/** ( L1 L2 -- L3 ) "l-divide"
Divides 32-bit longwords. On a 16-bit machine, **L/** is the same as **D/** (which is not included in the standard). On a 32-bit machine, **L/** is the same as **/**.

I haven't mentioned all the operators that are needed, but the rest of them are named in the obvious way. For instance, **L=** compares two 32-bit longwords for equality.

### Stack Manipulations

The "2" stack operators, such as **2SWAP** and **2DUP**, are unsatisfactory for manipulating 32-bit longwords. Such operators were originally intended for manipulating pairs of numbers, which are distinctly different from 32-bit longwords. I propose a set of 32-bit stack manipulation operators whose names begin with (you guessed it) the letter L. Examples are **LSWAP** and **LDUP**.

Mixing 32-bit numbers and 16-bit numbers on the stack poses problems. In a 32-bit system, all stack entries are 32 bits, so this is not too bad. On a 16-bit system, both 32-bit and 16-bit numbers may need to coexist on the stack. Currently, this is handled in an ad hoc fashion, using operators like **ROT** to separately manipulate the pieces of the numbers. Programs that do this are not portable to 32-bit machines (here I assume that we have decided against using 64-bit numbers). What we need is a set of operators for manipulating mixed stacks. The needed operators mostly duplicate existing functions, so we really don't need new capability, just new names! The new names will clearly specify the sizes of the operands.

**LDUP** ( L -- L L ) "l-dupe"
Duplicates a 32-bit longword. On a 16-bit machine, **LDUP** is equivalent to **2DUP**. On a 32-bit machine, **LDUP** is equivalent to **DUP**.

**LSWAP** ( L1 L2 -- L2 L1 ) "l-swap"
Exchanges 32-bit longwords. On a 16-bit machine, **LSWAP** is equivalent to **2SWAP**. On a 32-bit machine, **LSWAP** is equivalent to **SWAP**.

**LOVER** ( L1 L2 -- L1 L2 L1 ) "l-over" Copies a 32-bit longword over a 32-bit longword. On a 16-bit machine, **LOVER** is equivalent to **2OVER**. On a 32-bit machine, **LOVER** is equivalent to **OVER**.

```
CREATE MYARRAY  100 2* ALLOT
: FILLIT  ( -- )  100 0  DO  I   MYARRAY  I 2*  +  !   LOOP  ;
```

*Figure 1a.*

```
CREATE MYARRAY  100 /N * ALLOT
: FILLIT  ( -- )  100 0  DO  I   MYARRAY  I /N *  +  !   LOOP  ;
```

*Figure 1b.*

```
NA1+     ( addr -- addr+/n )          "n-a-one-plus"
LA1+     ( addr -- addr+/l )          "l-a-one-plus"
WA1+     ( addr -- addr+/w )          "w-a-one-plus"
CA1+     ( addr -- addr+/c )          "c-a-one-plus"
```

*Figure 2. Words to increment an address by the appropriate amount.*

```
/N*      ( n -- n*/n )                "per-n-times"
/L*      ( n -- n*/l )                "per-l-times"
/W*      ( n -- n*/w )                "per-w-times"
/C*      ( n -- n*/c )                "per-c-times"
```

*Figure 3. Words to scale by different sizes.*

```
NA+      ( addr n -- addr+n*/n )      "n-a-plus"
LA+      ( addr n -- addr+n*/l )      "l-a-plus"
WA+      ( addr n -- addr+n*/w )      "w-a-plus"
CA+      ( addr n -- addr+n*/c )      "c-a-plus"
```

*Figure 4. Words to index into arrays.*

**LDROP** ( L1 -- ) "l-drop"
Removes a 32-bit longword from the stack. On a 16-bit machine, **LDROP** is equivalent to **2DROP**. On a 32-bit machine, **LDROP** is equivalent to **DROP**.

**LROT** ( L1 L2 L3 -- L2 L3 L1 ) "l-rote"
Rotates 32-bit longwords. On a 16-bit machine, **LROT** is equivalent to **2ROT**. On a 32-bit machine, **LROT** is equivalent to **DROT**.

**LNSWAP** ( L n -- n L ) "l-n-swap"
Exchanges a 32-bit longword with a normal number. On a 16-bit machine, **LNSWAP** is equivalent to **ROT ROT**. On a 32-bit machine, **LNSWAP** is equivalent to **SWAP**.

**NLSWAP** ( n L -- L n ) "n-l-swap"
Exchanges a normal number with a 32-bit longword. On a 16-bit machine, **NLSWAP** is equivalent to **ROT**. On a 32-bit machine, **NLSWAP** is equivalent to **SWAP**.

**LNOVER** ( L n -- L n L ) "l-n-over"
Copies a 32-bit longword over a normal number. On a 16-bit machine, **LNOVER** is equivalent to **2 PICK 2 PICK**. On a 32-bit machine, **LNOVER** is equivalent to **OVER**.

**NLOVER** ( n L -- n L n ) "n-l-over"
Copies a normal number over a 32-bit longword. On a 16-bit machine, **NLOVER** is equivalent to **2 PICK**. On a 32-bit machine, **NLOVER** is equivalent to **OVER**.

**L>R** ( L -- ) "l-to-r"
Moves a 32-bit longword to the return stack.

**LR>** ( -- L ) "l-r-from"
Moves a 32-bit longword from the return stack.

**L>R** and **LR>** are provided to help with more complicated stack manipulations involving mixed stacks. (However, it is usually preferable to try to avoid complex stack gymnastics, instead.)

**Accessing Memory**
We need some words for accessing memory items of various sizes. We already have **C@**, **C!**, **2@**, **2!**, **@**, and **!**. We also need some words to access exactly 16 bits and exactly 32 bits, so we add these words:

**W@** ( adr -- 16b ) "w-fetch"
Fetches the 16-bit word at adr. On a 32-bit machine, the result is padded with zero to form a 32-bit normal number on the stack.

**<W@** ( adr -- 16b ) "signed-w-fetch"
Fetches the 16-bit word at adr. On a 32-bit machine, the result is sign-extended to form a signed, 32-bit normal number on the stack.

**W!** ( 16b adr -- ) "w-store"
Stores the 16-bit word at adr. On a 32-bit machine, the number "16b" is represented on the stack as the lower half of a 32-bit normal number, and only the lower 16 bits are stored at adr.

**L@** ( adr -- L ) "l-fetch"
Fetches the 32-bit longword at adr. On a 16-bit machine, the result is left on the stack as two 16-bit numbers, with the most-significant half on the top of the stack.

**L!** ( L adr -- ) "l-store"
Stores the 32-bit longword "L" at adr. On a 16-bit machine, the longword "L" is represented on the stack as two 16-bit numbers, with the most-significant half on the top of the stack.

**Type Conversion**
Some words for converting to and from 32-bit longwords:

**N->L** ( n -- L ) "n-to-l"
Converts a normal number to an unsignd 32-bit longword. On a 16-bit machine, **N->L** is equivalent to 0. Does nothing on a 32-bit machine.

**S->L** ( n -- L ) "signed-to-long"
Converts a normal number to a signed 32-bit longword. On a 16-bit machine, **S->L** is equivalent to **S->D**. Does nothing on a 32-bit machine.

**L->N** ( L -- n ) "l-to-n"
Converts a 32-bit longword to a normal number. On a 16-bit machine, **L->N** is equivalent to **DROP**. Does nothing on a 32-bit machine.

**Important Note**
It is neither necessary nor desirable to use the "L" operators all the time on a 32-bit system. Most of the time you don't really care whether a number is 16 bits or 32 bits. So you just use the normal operators like +, -, **DUP**, etc. The right time to use the "L" operators is when:

1. You require more than 16 bits, and
2. You want your code to run on both 16-bit and 32-bit machines.

In particular, I use the "L" operators when converting "double numbers" from 16-bit machines to 32-bit machines. Also remember that "double numbers" on 16-bit machines are not always the same as 32-bit longwords. Sometimes the "double number" operators are used to manipulate *pairs* of single numbers. Such uses must remain unchanged. A pair of numbers is still a pair of numbers, even on a 32-bit machine.

**The Name's the Thing**
The underlying problem is that Forth often uses the same name for different purposes. Examples: **2+** is used to add the number two, and to increment an address to the next word; **2DROP** is used to remove two single numbers from the stack, and to remove a 32-bit number from the stack. This happens to have worked in the past, because Forth just assumed that every machine in the world is a 16-bit, byte-addressed machine. (In fact, it didn't work on the Data General Nova, which is word addressed!) Different conceptual functions should have different names, even if the functions happen to have identical implementations on a particular machine.

That is why I am proposing new names without any new functions!

**Yes, But ...**
If we make all these new names for functions we already have, like

```
: LSWAP ( L1 L2 -- L2 L1 )
2SWAP ;
```

doesn't the code run slower? There are two answers:

1. It's not much slower.
2. There's a way to get the speed back.

To speed it up, see the article on synonyms (Yngve, Victor H. "Synonyms," *Forth Dimenions* VII/3). A synonym provides a new name for an existing word, with no run-time penalty. Basically, a synonym is an extra name for an existing word. When the compiler sees the new name, it compiles the compilation address of the existing word.

**Tips for Converting Programs**

When I convert a 16-bit program to a 32-bit machine, the first thing I do is search for all occurrences of the character 2. I look for the number two and for words that start with 2, such as **2DROP**. Many of these have to change. Usually, words like **2+** change to **NA1+**, and **2*** to **/N***, etc. For words like **2DROP**, you have to decide whether it is being used to drop a pair of normal numbers (in which case you leave it alone), or a 32-bit number (in which case it changes to **LDROP**).

The next thing I look for is words like **D+**, **D-**, etc., changing them to **L+**,**L-**, etc.

Sometimes the word 0 is used to convert an unsigned normal number to a longword. This has to change to **N->L**.

Finally, I look for cases where stack manipulations like **ROT** are used to swap mixed 32-bit and 16-bit numbers. These change to words like **NLSWAP**. These

are not as hard to find as you might guess, because they usually occur just before a 32-bit operator, so you should keep an eye out for them while looking for 2, D+, 0, etc.

Since each of the words proposed here can be implemented so easily, it is easy to add them to the program as you need them. I usually group their definitions together at the beginning of the source code, to make them easier to find.

Sometimes you may find you need a mixed 16-bit/32-bit operator not mentioned here. I recommend that you pick a name starting with either "NL" (if the longword is on the top of the stack) or "LN" (if the normal number is on top).

**Further Reading**

This wordset was originally proposed in two earlier papers.


1. Bradley, Mitch and Sebok, Bill. "Compatible Forth on a 32-bit Machine," *The Journal of Forth Application and Research*, Vol. 2 No. 4, 1984.

2. Bradley, Mitch and Sebok, Bill. "Extended Addressing Wordset," Working Group Report, *Proceedings of the 1984 Rochester Forth Conference*.


These papers describe some other words not mentioned here, including names for words to perform extended addressing on 16-bit machines. Vol. 2 No. 4 of *The Journal of Forth Application and Research* contains several papers describing other aspects of 32-bit machines.


*Mitch Bradley is the owner of Bradley Forthware, vendor of the 32-bit "Forthmacs," a very complete Forth-83 environment for the Atari ST.*


suite de la page 22

en question

2) creation d'un assembleur FORTH implante dans TURBO-Forth. On l'appellerai ASSEMBLF32 par exemple et toutes les instructions CODE..END-CODE redirigees sur ce nouveau vocabulaire. Une sequence d'assemblage en assembleur F32 genere un code executable 'virtuel' dans un segment memoire approprie. Le contenu de ce segment pourra etre sauvegarde sur disque.

3) definition d'un simulateur, construit sur le principe de celui de Mr DUMUR (voir JEDI 44) mais tenant compte d'un systeme F32 virtuel (memory mapping, interfaces, buffers, etc...) permettant une execution pas-a-pas ou 'temps-reel'.

Bien entendu, ce systeme virtuel ne serait pas aussi performant que le vrai systeme F32, mais permettrait de tester des programmes 'grandeur nature' alors que le F32 n'existe pas encore.

Pour ma part, je me propose de diffuser a la demande des volontaires interesses par le projet F32, le simulateur de Mr DUMUR sur disquette, ceci pour ceux qui n'arrivent pas a le telecharger depuis SAM*JEDI. Deposez votre demande, en precisant nom et adresse, dans la BAL JEDI.

---

De PILVERDIER Du 11.07.88 A 15h05
DANS F-PACK :
LE MOT ?10PWR NE RENVOIE PAS LA VALEUR A LAQUELLE ON POURRAIT S'ATTENDRE. EST-CE NORMAL VU SON UTILISATION?